Branch and Bound Algorithm for Finding the Maximum Clique Problem

Mochamad Suyudi, Sukono
Department of Mathematics, Faculty of Mathematics and Natural Sciences
Universitas Padjadjaran, Indonesia
moch.suyudi@gmail.com, sukono@unpad.ac.id

Mustafa Mamat
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin
Tembila Campus, 2200 Besut, Terengganu, Malaysia
musmat567@gmail.com

Abdul Talib Bon
Department of Production and Operations,
University Tun Hussein Onn Malaysia, Johor, Malaysia
talibon@gmail.com

Abstract
We present a branch and bound algorithm for the maximum clique problem in arbitrary graphs. The main part of the algorithm consists in the determination of upper bounds by graph colorings. Using a modification of a known graph coloring method called heuristic greedy we simultaneously derive lower and upper bounds for the clique number.

Keywords:
Maximum Clique problem, Exact Algorithms, Approximation Algorithms, Heuristic

1. Introduction
Clique problem, refers to the problem of finding a complete set of sub graphs (cliques) in a graph, i.e. a set of elements that are mutually connected. An undirected graph G = (V, E) has V and E $\subseteq$ V $\times$ V as a set of vertices and a set of edges, respectively. A clique is a set of vertices C $\subseteq$ V that there is an edge for every of pair of vertices (Bomze,1999).

Maximum clique and maximal clique are different. A maximal clique isn't subset any other clique, but maximum clique is a maximal clique that have most number of vertices.So a maximum clique is maximal but a maximal clique might be maximum clique (Babel,1990).

If P $\neq$ NP for any $\varepsilon > 0$ there is no multinomialtime algorithm for estimation the maximum clique within a factor of $n^{1-\varepsilon^3}$. This feature makes MCP difficult to solve. Bomze provided an overview of the algorithms (such as Local Search Heuristics, Neural networks, Genetic algorithms and Tabu search) and applications of the MCP's types (such as undirected, unweighted and weighted) (Bomze,1999).

Most of the known exact algorithms are based on Branch-and-bound (Suyudi et al., 2016). This algorithms, first recursively expands clique size using addition of vertices from a candidate set. Then, by pruning based on applying a simple constraint, separates unused vertices of the search tree. This algorithm loses its effectiveness when the number of vertices increases (Beigel,1999). Bit Parallelism improved these algorithms (Bomze,1999). Another solution for solving MCP is using genetic algorithm (GA) that using a simple GA is not suitable to solve MCP. To solve this issue, GA needs to be combined with other techniques to improve performance (Bron and Kerbosh,1973). In heuristic genetic algorithm (HGA), a meta-heuristic greedy approach, first expands subsets of vertices by adding randomly picked vertices, then it decrease it to a single clique, then expands it to the largest clique (Carmo, 2012; Carraghan,1990). EA/G Algorithm is accompanied with a mutation operator that uses statistical information relating to local information (Babel,1990). Ant-Clique algorithm is a solution to MCP using ACO algorithm in which a greedy sequential heuristics creates maximum clique by frequent addition of vertices to partial cliques (Cook, 1971). A two-

step IEA/PTS algorithm is based on the Tabu-Search that uses a two-step evolutionary strategy involving discovery of the issue and finding a solution (Dirac,1952; Fahle, 2002). Applications of maximum clique are study of user behavior on ecommerce, coding theory and computer vision (Garey and Johnson,1979).

In next section, the greedy approach to find the maximum clique is presented and simulation results are investigated. In greedy approach, which is based on the maximum number of adjacent vertices, by removing vertices that are of less chance to be in maximum clique, the solution is obtained. Obtained solution from next section will be used as initial countries in imperialist competitive. ICA, by switching vertices and adding new vertices to the initial countries, leads to the optimal solution. In last section, results and conclusions are discussed.

2. Notations and Definitions

Given an arbitrary *simple undirected graph* $G = (V, E)$ without loops and multiple edges. The order of a graph is the number of vertices $n=|V|$. A graph's size is the number of edges $m=|E|$.

A *complement* of graph G is the graph $\bar{G} = (V, \bar{E})$ where $\bar{E} = \{ (v, u) \mid v, u \in V, v \neq u, \text{and } (v, u) \notin E \}$.

A *clique* C is a set of pair wise adjacent vertices of the graph. The *maximum clique problem*(MCP) is to find a clique of maximum cardinality in a graph G.

An *independent set (stable set, vertex packing)* I is a set of pair wise nonadjacent vertices of the graph. The *maximum independent set* (MIS) problem is to find an independent set of maximum cardinality in a graph G.

A *vertex cover* C is a subset of V such that every edge $(v, u) \in E$ is incident to at least one vertex in S. The *minimum vertex cover* (MVC) problem is to find a vertex cover of minimum cardinality in a graph G.

It is easy to see that C is a clique in a graph $G = (V, E)$ if and only if $V - C$ is a vertex cover in the complement graph $\bar{G} = (V, \bar{E})$, and if and only if C is an independent set of $\bar{G}$. Thus, the maximum clique problem, the vertex cover problem and the maximum independent set problem are equivalent.

In addition, they are all NP—complete, which means that unless P= NP there exists no algorithm that can solve this problems in time polynomial to the order of the input graph.

For $k \geq 1$ a *k-coloring* of G is a mapping φ of $V(G)$ into the (color-) set $\{1, \ldots, k\}$ satisfying $\varphi(v) \neq \varphi(u)$ for any adjacent vertices $v, u \in V$. A graph which admits a k-coloring is called *k-colorable*.

The *chromatic number* $\chi(G)$ of a non-empty graph G is the smallest integer k for which G is k-colorable. If $\chi(G) = k$ then G is called *k-chromatic.*

Note that $H \subseteq G$ implies $\chi(H) \leq \chi(G)$.

Proposition 2.1 *For every graph G the clique number* $\omega(G)$ *is a lower bound on the chromatic number* $\chi(G)$

$\omega(G) \leq \chi(G)$.

3. Maximum Clique Algorithm Complexity

A lot of effort has been devoted in last decades to develop faster and faster exact algorithms MC problem. Let us remember the main achieved results. The Bron–Kerbosch (1973), algorithm is a recursive backtracking procedure that augments a candidate clique by considering one vertex at a time, either adding it to the candidate clique or to a set of excluded vertices that cannot be in the clique but must have some non-neighbor in the eventual clique.

Tomita (2007), presented a depth-first search algorithm for generating all maximal cliques of an undirected graph, in which pruning methods are employed as in the Bron–Kerbosch algorithm and proved its worst-case time complexity $\boldsymbol{O(3^{n/3})}$ or $\boldsymbol{O(2^{0.528n})}$.

$\boldsymbol{O(2^{n/3})}$. On the other hand, finding the maximum clique of a graph does not require to actually examine all of its maximal cliques. Along the search among the maximal cliques of the graph, some non-maximal cliques can be discarded as soon as they are identified as not contained in a clique larger than another already known. Tarjan and Trojanowski(1976) proposed such algorithm with worst case running time of $\boldsymbol{O(2^{n/3})}$.

$\boldsymbol{O(2^{0.304n})}$. Jian(1986) improved algorithm complexity to $O(2^{0.304n})$.

$\boldsymbol{O(2^{0.296n})}$ **and** $\boldsymbol{O(2^{0.276n})}$. In 1986 Robson proposed two versions of a new but similar algorithm: the first version runs in polynomial space in time $O(2^{0.296n})$ and the second version, which used exponential space, needs $O(2^{0.276n})$.

$\boldsymbol{O(2^{n/4})}$. In 2001 Robsonreduced this bound to $O(2^{n/4})$ and this bound remaining the best known.

$\boldsymbol{\Omega(2^{0.166n})}$ **and** $\boldsymbol{O(2^{0.288n})}$. Fomin(2006) proposed $O(2^{0.288n})$ algorithm and simultaneously he showed $\Omega(2^{0.166n})$ lower bound on the worst-case time complexity of his MIS algorithm. But unfortunately no experimental results/tests of this algorithm are known.

Group of "practical" algorithms commonly use the branch and bound method. The key issues in a branch and bound algorithm for the MCP are [Bomze,199]:

1. How to find a good lower bound, i.e. a clique of large order?
2. How to find a good upper bound on the order of maximum clique?
3. How to branch, i.e. break a problem into smaller subproblems?

One of the most important contributions in the 1980's on practical algorithms for the MCP is due to Balas and Yu(1986). They proposed to use on the second phase(upper bound) a well known fact that the chromatic number of a graph is always bigger or equal to the order of this graph clique number $\omega(G) \leq \chi(G)$.
Later appeared a lot of algorithms using different heuristic vertex-coloring on the second phase Babel(1990), Wood(1997), Tomita(2007), Fahle(2002), Regin([2003), Konc(2007), Kumlander(2005), Chu-Min Li(2010) and etc..

The problem of vertex coloring is NP—complete too Karp (1972) and therefore MCP algorithms use heuristic vertex coloring techniques(DSATUR, GREEDY and other), for example see Klotz (2002), Radin (2000), and de Werra (1990) . Such MCP algorithms demonstrate quite good results on the random graphs and more worse results on the special benchmarks, such as DIMACS, BHOSLIB or Sloane.

Recently appeared a lot of good surveys (Segundo (2011),Carmo(2012)] in which different MCP algorithms have been compared. Carmo(2012) proposed to consider the following:
"Explaining the gap between the disheartening worst case estimates and what has actually already been achieved in practice seems to be an interesting challenge."

With this long term goal in mind, we will show that MC algorithms using different heuristic vertex-coloring cannot run better than $\Omega(2^{0.2n})$. Notice that we are concerned with lower bound on the complexity of a particular class of MCP algorithms, and not with lower bounds on the complexity of an Maximum Clique Problem.

4. Branch and Bound Algorithm

In this section it will discuss determine the upper and lower bound, and branching procedure as follows. A well-known exact algorithm (denoted by EA; enumerative algorithm) is developed by **Carraghan and Pardalos (**1990) which is shown in Algorithm-2. Despite its simplicity, this algorithm constitutes an important step for exact solving of the MCP and provides the basis for many later improved exact clique algorithms. The functioning of this algorithm is discussed below in detail (Suyudi et al., 2016).

Algorithm 2: Branch &Bound algorithm

```
Function clique (U; C)
1:    if |U| =0 then
2:          if |C| >| C*| then
3:             |C*|:=|C|
4:           New record; save it.
5:       end if
6:       return
7:     end if
8:  while U ≠Ø do
9:        if |C| + |U| < |C*| then
10:             return
11:      end if
12:    i:=min{j | vj∈U}
13:   U:=U-{v}
14:   clique(U ∩ N(vi); C ∪i)
15:   end while
16:   return
function old
17:   C*:=Φ
18:   clique (V; Φ)
19:   return
```

Vertex set V is first ordered and one by one vertex is explored. The vertices of G is ordered into a list L = (v_1, v_2..., v_n) where vn is the vertex of minimum degree in G, v_{n-1} is the vertex of minimum degree in G-{v_n}, and v_{n-2} is the vertex of minimum degree in G-{v_n,v_{n-1}}, and so on. Once the maximum clique induced on a particular vertex (v_i)

is found, it (v_i) is removed from the ordered list. In other word, every time N(v) is found in right side from itself of the ordered list. Each vertex of U is connected to all the vertices of C, i.e. any vertex v of U can be added to C to obtain a larger clique C'= C ∪ {v}. The pruning is done when the set U (current candidate set) becomes so small that even if all vertices in U would be added to the C (current local clique size), the size of that clique would not exceed that of the largest clique found previously.

Osterg'ard (2002) proposed a better heuristic algorithm (it is named here REA, i.e. reverse enumerative algorithm) with reverse ordering as in algorithm-2, and improved the upper bound of the EA algorithm described above. It uses an additional memory to store the clique size induced on each of the vertex and latter it is used while pruning the branch. The algorithm remembers the maximum clique found for each vertex previously into a special array *b.* So *b*[*i*] is the maximum clique for the *i*-th vertex while searching backward. This number is used later as: if we search for a clique of size greater than |C*|, then the search on vi is pruned if *vi* is going to be the (*j* + 1)-th vertex in C and $j+ b[i] \leq |C^*|$.

To estimate the upper bound of the maximum clique, graph coloring techniques are also applied to the subgraph induced by the candidate set U. This is based on a general fact that if a graph can be colored with k colors, then the maximum clique in this graph must be smaller or equal to k. Using color classes instead of b[i] (mentioned above) improves the upper bound and consequently reduces the size of the search tree. In addition, vertex coloring is also a NP-hard problem and may be expensive so, a greedy method may be used for coloring the vertex set U during the search process. The following (next two) algorithms of exact solution, unlike EA and REA, are based on color based pruning.

Deniss Kumlander(2006) proposes a better heuristic based vertex coloring and backtracking for MCP. The algorithm works like REA, mentioned above, but the pruning condition is different. Initially vertices are sorted by color classes obtained by a heuristic vertex coloring algorithm, i.e. $V = \{C_n, C_{n-1}, ..., , C_1\}$, where C_i is a set of vertices with I, i.e. *i*-th color class. First of all cliques that could be built on vertices in *C*1 are explored. Then on vertices of C_1 and C_2, i.e. of the first and second color classes, and so forth. In other word; at the *i*-th step all cliques can that contain vertices of $\{C_i, C_{i-1}, ..., , C_1\}$, are explored. The algorithm remembers the maximum clique found for each for each color class into a special array *b.* So *b*[*i*] is the maximum clique for a subgraph formed by $\{C_i, C_{i-1}, ..., C_1\}$ vertices while searching backward. This is used later for pruning the branch; if max clique size found so far is $|C^*|$, then if *v*i is going to be (*j* + 1)-th vertex in C and it belongs to the *k*-th colour class and $j+ b[k] \leq |C^*|$, then the branch is pruned.

An improved version of greedy coloring based algorithm is proposed as **MCS** Tomita (2010), which uses a recoloring strategy using greedy approximate coloring procedure and outperform its predecessors in Tomita (2003;2007). Vertices of U are sorted in an ascending order with respect to the color number. Then, at each search step of the algorithm, MCS selects a vertex v ∈ U in reverse order (the last vertex in the reordered set U belongs to the highest color class) and the color number associated with each vertex becomes an upper bound for the maximum clique in the remaining subgraph in the current branch to be searched. The basic idea of pruning in MCS is that if a vertex v ∈ U with color number Kv< (|C*| - |C|), then the vertex v needs not to be searched from. The number of vertices to be searched can be further reduced, as to move vertex v with color >(|C*| - |C|) to other color class less than (|C*| - |C|) in number. This is how number of vertices to be searched in the candidate set U is reduced.

A constraint programming (CP) based B&B algorithm is proposed in Kluwer(2003), Jean-Charles Regin (2003), it provides a heuristic to filter the vertices which is not going to lead to a better clique size. The approach that can be viewed as an adaptation and a generalization of the Bron & Kerbosh's (1973) ideas for enumerating the maximal cliques of a graph. The upper bound of clique size in every branching step is computed based on a matching algorithm rather than a coloring algorithm. For each vertex v in U, the upper bound of clique size roughly corresponds to the number of vertices in N(v) minus matching number in the subgraph (its complement graph) induced by N(v). If this upper bound plus the current clique size is smaller than the maximum clique obtained in previous all branching steps, then v can be removed from U. Concluding the section of exact algorithm for MCP,

a simple greedy algorithm for MAXCLIQUE is illustrated in Figure 1. Greedy algorithms are frequently used in practice for their simplicity of implementation and better efficiency. In greedy heuristics, decisions on vertex to be added in or moved out are usually based on some static information associated with the vertices in the candidate set like their degrees. Several improvements to the static greedy heuristics have been proposed in the literature. For instance QUALEX-MS in Busygin(2006) and DAGS in Grosso(2006).

Local search is a sophisticated way of using greedy approach. However, greedy algorithms can easily fall into the local optima due to their short-sighted nature. Several improvements to the greedy heuristics Bahadur (2002), Etsuji Tomita (2011), Balasundaram (2011) have been proposed in the literature.

Although most algorithms have been empirically evaluated on benchmark instances from the Second DIMACS Challenge but, somewhat unsurprisingly, there is no single best algorithm. Nevertheless, there are few heuristic MAX-CLIQUE algorithms, described in the subsequent sections that achieved state-of-the-art performance.

Procedure Greedy_Add(*v*∈*V)*
1. Set C = C∪*{v};*
2. set N(C)=N(v)
2. ***while*** *N(C)* ≠ Ø ***do***
3. Select i ∈*N(C)*
Such that |N (i) ∩ *N(C)| is maximum;*
4. Set C = C ∪ *{i };*
5. ***end while****;*
6. ***return C****; //C is clique induced on v*

Figure 1: A simple greedy procedure called in Clique improvement phase

Procedure Greedy_swap_Move (*v*∈*V)*
1. Select i ∈*N1(C)*
Such that |N (i) ∩ *N(C)| is maximum or at random*
2. Set C = C ∪ *{i};*
3. Remove/drop vertex v from C, which is not adjacent to i
4. Store the removed vertex v in tabu list,
// vertices in tabu list are prohibited till a
specified no. of iteration.
4. Update N(C) & N1(C), excluding the vertices in tabu list.

Figure 2: A simple greedy procedure called for Plateau search

Procedure to determine the upper, lower bound, and branching as follows.
To find a lower bound (LB) on the size of the maximum clique, a simple greedy clique-finding heuristic can be implemented as follows on the graph *G =(V,E)*:

1. Let $T = \phi$.
2. Let *v* be the vertex in *G* but not in *T* with maximum degree. Add *v* to *T* .
3. Delete all vertices not adjacent to *v* from *V* .
4. If *V* is empty, stop. Otherwise, go back to step 2.

Basically, we add to the clique *T* the vertex with the largest degree that is connected to all the vertices already in the clique. The cardinality of the resulting set *T* is a clique in the graph *G* and is, therefore, a lower bound on $\omega(G)$, the size of the maximum clique in *G*. Figure 3 illustrates this heuristic.
To find an upper bound (UB) on the size of the maximum clique, a simple greedy coloring heuristic can be implemented as follows on the graph *G = (V,E)*:

1. Color vertex v_1 with color c_1.
2. For each successive vertex v_i , choose the lowest numbered color that does not produce an invalid coloring (i.e. such that no other previously-colored vertex adjacent to *vi* shares the same color).

The result of this heuristic is a valid coloring of *G*, which is an upper bound on $\omega(G)$, the size of the maximum clique in *G*.

Once the graph theory basics have been covered, we can solve the maximum clique problem on any reasonably-sized graph (i.e. one that can be drawn on a blackboard) by using the method of branch-and bound. We can write the entire solution method in their notebooks, often on just one page. This is, of course, not the case when using traditional integer programs where each node in the branch-and-bound tree requires solving a linear program, either manually or on a computer.

The following branching procedure:

1. Given graph $G = (V,E)$, choose the branching vertex $v \in V$, where v is any vertex not connected to all the other vertices in G. If no such v exists, then G is a clique. Otherwise, go to 2.
2. Create subgraphs G' and G" from G as follows:
   - G' is the subgraph of G induced by vertices $V - \{v\}$, i.e. G' is formed by deleting the vertex v (and its adjacent edges) from G.
   - G" is the subgraph of G induced by vertices v in $N\{v\}$ where $N\{v\}$ represents the neighbors of vertex v, i.e. G" is formed by keeping only the vertex v and all vertices adjacent to it in G.

The maximum degree node either is or is not a member of a maximum clique. Because G' deletes only the maximum degree node from its parent graph and G" keeps the maximum degree node and all its neighbors, the way in which the subgraphs G' and G" are created ensures that a maximum clique in G will still exist in one or both of the subgraphs. For more detail, see Strickland (2002), Mochamad Suyudi(2016). Thus, maximum cliques are not destroyed in this process, but the size of the graphs at each branching vertex decreases with every iteration.

5. Calculation Result of Branch and Bound

Now that the branching procedure and calculation boundary has been explained, the entire branch and bound algorithm can be further illustrated. Figure 2 shows the sequence of branching processes vertices, using upper and lower bounds to prune the branches, and the clique and coloring searched using heuristic described above. Clique is outlined in black, and coloring are shown generated using heuristic greedy clique. Clique alternatives can be sought when different options, to the maximum degree of a node is made (in the case of ties), and color alternatives may be sought if the order of vertices different from the one shown above.

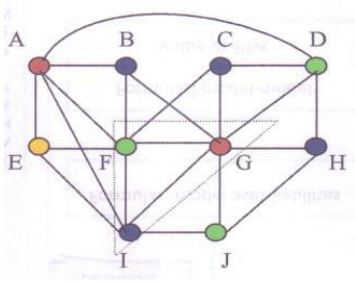

Figure 3. The sequence of branching processes vertices

Select vertex G, because vertex G has degree of vertex larger than other vertices.

Branching on the vertices *G*

Step 1. *Clique* = {F,G,I}; LB= $\omega(G) = 3$ and Color = UB = $\chi(G) = 4$

Step 2. Remove vertices G, obtained Clique = {A, E, F, I}, LB = $\omega(G) = 4$. Color = UB = $\chi(G) = 4$

Step 3. Remove all vertices that are not adjasen with G, obtained Clique = {F,G,I}; LB = $\omega(G) = 3$. Color = UB = $\chi(G) = 3$.

Branching on the vertices F

Step 4. Remove vertices F in Step 2, obtained Clique={A, E, I}; LB = $\omega(G)$=3.Color = UB = $\chi(G)$=4

Step 5. Remove all vertices that are not adjasen with vertices F in step 2, obtained Clique = {A, E, F, I}; LB = $\omega(G) = 4$. Color = UB = $\chi(G) = 4$.

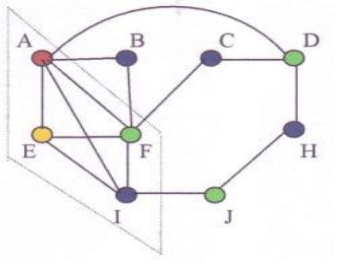

Figure 4. Step 2, Clique={A,E,F,I), LB= $\omega(G)$ =4. Color=UB=$\chi(G)$=4

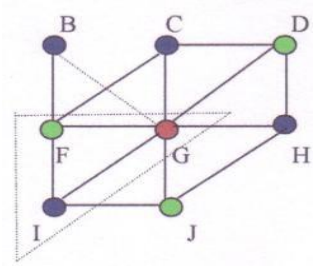

Figure 5. Step 3, Clique={F,G,I}, LB= $\omega(G)$=3. Color=UB=$\chi(G)$=4

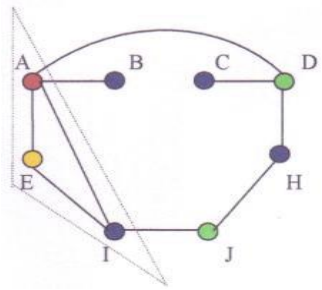

Figure 6. Step 4: Clique={A,E,I}, LB= $\omega(G)$ =3. Color=UB=$\chi(G)$=3

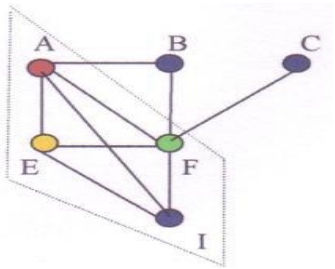

Figure 7. Step 5: Clique={A,C,D,G}, LB= $\omega(G)$=4. Color = UB = $\chi(G)$=4. Optimum, where $4 \leq \omega(G) \leq \chi(G) = 4$.

From Figure 2 to Figure 6, or from step 1 to step 5, is an example of the whole procedure using branch and bound method in solving the maximum clique problem.

6. Conclusion

Based on the review of various algorithms, in the view of exact solution of MCP, further enhancement may be done by applying better coloring algorithm to estimate upper bound. In the view of local search technique, we can see that hardly a algorithm dominates on all other algorithm, it is because of diverse structure of graphs. One possible way to overcome this deficiency may be to incorporate multiple search operators within a single algorithm and incorporating dynamic capability to decide the most permissible operators to be triggered during the search process. The use of branch and bound method is advantageous, because it provides an alternative perspective to solve the maximum clique problem, as well as the optimal method of solution requires no software optimization. Branch and bound method has been used to the search the maximum clique problem. To determine the chromatic number (G) of a graph G has been sought by the greedy heuristic, chromatic number (G) is the upper bound (UB) on G and by branching procedure obtained lower bound (LB) is the maximum clique (G) on G. Results of the maximum clique the search also shows that the maximum clique (G) is less than or equal premises chromatic number (G).

Acknowledgements

Thank you for the program of the Academic Leadership Grant (ALG), Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran (Indonesia), which has been providing facilities to conduct a research and publication.

References

Babel, L. and Tinhofer, G.. A branch and bound algorithm for the maximum clique problem, *Meth. Oper. Res. 34:207–217.*1990

Balas, E. and Yu C.S. Finding a Maximum Clique in an Arbitrary Graph. *SIAM Journal Computing, 14(4:1054–1068.* 1986.

Beigel, R. Finding maximum independent sets in sparse and general graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA):856–857.*1999

BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring), http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graphbenchmarks.htm

Bomze, I.M., Budinich, M., Pardalos, P. M., and Pelillo, M. The maximum clique problem, *Handbook of Combinatorial Optimization*, **4**, Kluwer Academic Publishers, 1–74.1999

Bron, C. and Kerbosh, J. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM , 16(9):575-577.* 1973

Carmo, R. and Zuge, A. Branch and bound algorithms for the maximum clique problem under a unified framework. *J. Braz. Comp. Soc., 18(2):137–151.* 2012

Carraghan, R. and Pardalos, P.M. An exact algorithm for the maximum clique problem. *Operations Research Letters, 9:375–382.*1990

Cook, S. A. The complexity of theorem-proving procedures, Proc. 3rd ACM Symposium on Theory of Computing, 151–158. 1971

DIMACS clique instances, http://cs.hbg.psu.edu/txn131/clique.html

Dirac, G.A. A property of 4-chromatic graphs and some remarks on critical graphs, *J. London Math. SOC. 27 (1952):85-92.*1952

Fahle, T. Simple and Fast: Improving a Branch-and-Bound Algorithm for Maximum Clique. *In Proceedings ESA 2002, LNCS 2461:485–498.*2002

Fomin, F.V., Grandoni, F., and Kratsch, D. Measure and Conquer: A Simple O(20.288^n) Independent Set Algorithm. *Proceedings of SODA 2006, ACM and SIAM:508–519.* 2006

Garey, M.R. and Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness. *WH Freeman & Co.*1979

Jian, T. An O(20.304^n) algorithm for solving maximum independent set problem. *IEEE Transactions on Computers, 35(9):847–851.*1986.

Li, C.M. and Quan, Z. Combining graph structure exploitation and propositional reasoning for the maximum clique problem. *22-nd International Conference on Tools With Artificial Intelligence:344–351.*2010

Karp, R. M. Reducibility among combinatorial problems, *in Miller, R. E.; Thatcher, J. W., Complexity of Computer Computations, New York: Plenum, 85–103.*1972

Klotz, W. Graph coloring algorithms. *Tech. Rep. Mathematik-Bericht 5, Clausthal University of Technology, Clausthal, Germany.*2002.

Konc, J. and Janežič, D. An improved branch and bound algorithm for the maximum clique problem. *MATCH Communications in Mathematical and Computer Chemistry, 58:569–590.*2007

Knuth, D.E. The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams. *Addison-Wesley Professional, pp. 272.*2009

Kumlander, D. Some Practical Algorithms to Solve The Maximum Clique Problem, *Diss. ; Tallinn, Techn. Univ., Thesis On Informatics and System Engineering C26.*2005

Ostergard, P.R.J. A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics, 120: 197-207.* 2002

Pardalos, P.M. and Xue, J. The Maximum Clique Problem. *Journal of Global Optimization, 4:301–324.*1994

Radin, A. *Graph Coloring Heuristics form Investigation of Smallest Hard to Color Graphs. MS Thesis Rochester Institute of Technology Computer Science Department May 16, 2000.*

Regin, J.C. Solving the maximum clique problem with constraint programming, *Proceedings of CPAIOR'03:634-648.*2003.

Robson, M.J. Algorithms for maximum independent sets. *J. of Algorithms, 7:425-440.*1986.

Robson, J.M. Finding a maximum independent set in time *O*(2n/4). *Technical Report 1251-01, LaBRI, Universite Bordeaux I.*2001.

Segundo, P.S., Matia, F., Rodriguez-Losada, D., Hernando, M. An improved bit parallel exact maximum clique algorithm. *Optimization Letters :*2011.

Sloane, N.J.A. Challenge Problems: Independent Sets in Graphs, *Information Sciences Research Center*, http://neilsloane.com/doc/graphs.html

Suyudi, M., Mamat, M., & Sukono. An efficient approach for traveling salesman problem solution with branch-and-bound. Paper presented at the *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 2016, 543-546. Retrieved from www.scopus.com

Tarjan, R. and Trojanowski, A. Finding a maximum independent set. *SIAM Journal on Computing, 6: 537-546.*1977.

Tomita, E. and Seki, T. An efficient branch-and-bound algorithm for finding a maximum clique, *Lecture Notes in Computer Science 2631:278-289.*2003.

Tomita, E., Tanaka, A., and Takahashi, H. *The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science, 363:28–42.*2006.

Tomita, E. and Kameda, T. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Glob. Optim., 37:95–111.*2007

Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., and Wakatsuki, M. *A simple and faster branchand-bound algorithm for finding maximum clique*. In *WALCOM 2010 : 191–203.*2010

Werra, de. D. Heuristics for Graphs Coloring, *Computational Graph Theory, Comput. Suppl. 7, Springer, Vienna (1990), 191-208.*1990

Woeginger, G.J. Exact algorithms for NP-hard problems: A survey. In: "Combinatorial Optimization - Eureka! You shrink!". M. Juenger, G. Reinelt and G. Rinaldi (eds.). LNCS 2570, Springer, 2003, 185-207.2003

Wood, D.R. An algorithm for finding a maximum clique in a graph. *Operations Research Letters, 21:211–217.* 1997

Biographies

Mochamad Suyudi, is a lecturer at the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran. Bachelor in Mathematics at the Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, and Master in Mathematics at the Faculty of Mathematics and Natural Sciences, Universitas Gajah Mada. Currently pursuing Ph.D. program in the field of Graphs at Universiti Sultan Zainal Abidin(UNISZA) Malaysia Terengganu.

Sukono is a lecturer in the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran. Currently serves as Head of Master's Program in Mathematics, the field of applied mathematics, with a field of concentration of financial mathematics and actuarial sciences.

Abdul Talib Bon is a professor of Production and Operations Management in the Faculty of Technology Management and Business at the Universiti Tun Hussein Onn Malaysia since 1999. He has a PhD in Computer Science, which he obtained from the Universite de La Rochelle, France in the year 2008. His doctoral thesis was on topic Process Quality Improvement on Beltline Moulding Manufacturing. He studied Business Administration in the Universiti Kebangsaan Malaysia for which he was awarded the MBA in the year 1998. He's bachelor degree and diploma in Mechanical Engineering which his obtained from the Universiti Teknologi Malaysia. He received his postgraduate certificate in Mechatronics and Robotics from Carlisle, United Kingdom in 1997. He had published more 150 International Proceedings and International Journals and 8 books. He is a member of MSORSM, IIF, IEOM, IIE, INFORMS, TAM and MIM.

Mustafa Mamat is a lecturerin the Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Malaysia. Currently serves as Dean of Graduate School Universiti Sultan Zainal Abidin, Terengganu, Malaysia. The field of applied mathematics, with a field of concentration of optimization.