Proc. Estonian Acad. Sci. Phys. Math., 2005, **54**, 2, 79–86
https://doi.org/10.3176/phys.math.2005.2.02

Problems of optimization: an exact algorithm for finding a maximum clique optimized for dense graphs

Deniss Kumlander

Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia; kalkin@solo.ee

Received 9 October 2003, in revised form 21 December 2004

Abstract. An algorithm for the maximum clique problem on arbitrary undirected graphs is described. The algorithm presumes, as has been proved, that vertices from an independent set cannot be included into the same maximum clique. The independent sets are obtained from a heuristic vertex-colouring, where each set constitutes a colour class. The colour classes are then used to prune branches of the maximum clique search tree. Computational results show that the algorithm performs better than those published earlier, showing a substantial improvement with dense graphs. Moreover, the new algorithm is easy to implement.

Key words: maximum clique, independent set, branch-and-bound algorithm.

1. INTRODUCTION

Let $G = (V, E)$ be an undirected graph, where V is the set of vertices and E is the set of edges. A clique is a complete subgraph of G, i.e. its vertices are pairwise adjacent. The *maximum clique problem* is a problem of finding the maximum complete subgraph of G, i.e. a set of vertices from G that are pairwise adjacent. An *independent set* is a set of vertices that are pairwise nonadjacent. A graph-colouring problem is defined to be assignment of colour to graph's vertices so that no pair of adjacent vertices will share identical colours. All of these problems are computationally equivalent, in other words, each of them can be transformed to any other.

These problems are NP-hard on general graphs [1]; no polynomial time algorithms are expected to be found. Great interest is being shown in developing a fast exact algorithm for instances with a reasonable number of vertices, since it can be used in several important practical applications. Examples are efficient register allocation [2], on-line bin-stretching [3], scheduling of parallel jobs [4–6], and overcoming failures in distributed computing [7–9]. All of the problems mentioned originate from the programming theory: “Linearity tests” and “Probabilistically checkable proofs” from the programming logic [10], “Problem of finding the optimum replacement” in the program construction [10], and other sources.

Many papers have been published since the early 1970s, presenting an algorithm for the maximum clique problem. A very simple effective algorithm was proposed by Carraghan and Pardalos [11]. This algorithm was used as a benchmark in the Second DIMACS Implementation Challenge [12], since it is reported to be the fastest one. We propose here a new algorithm that refines the Carraghan and Pardalos algorithm. Initially, the new algorithm was designed to be optimal on dense graphs, however, as a rule, it works better than other algorithms. The following section contains a description of the new algorithm.

2. DESCRIPTION OF THE ALGORITHM

First, we will find a vertex-colouring with the help of any heuristic algorithm, for example, in a greedy manner. We determine colour classes one by one as long as uncoloured vertices exist. The vertices are re-sorted in the order they are added into colour classes. This order affects the performance of the algorithm in finding the maximum clique and is therefore very important.

Definition 1. *A colour class is called existing on a subgraph* G_p *if any vertex from this colour class belongs to the subgraph* G_p.

Definition 2. *The degree of a subgraph* G_p *equals the number of colour classes existing on that subgraph*.

A notation of the depth and the pruning formula are crucial to the understanding of the algorithm. Basically, at depth 1 we have all vertices, i.e. $G_1 \equiv G.$ We will expand all vertices of a subgraph so that a vertex is deleted from the subgraph after it is expanded. Suppose we expand vertex v_1. At depth 2, we consider all vertices adjacent to v_1 among the vertices at the previous depth, i.e. belonging to G_1. These vertices form a subgraph G_2. At depth 3, we consider all vertices (that are at depth 2) adjacent to the vertex expanded at depth 2, etc.

Let us say that G_d is a subgraph of G at depth d, which contains the following vertices: $V_d = (v_{d,1}, \ldots, v_{d,m})$. Let $v_{d,1}$ be the vertex we are currently expanding at depth d. Then a subgraph at depth $d+1$ is

$$G_{d+1} = (V_{d+1}, E),$$

where $V_{d+1} = (v_{d+1,1}, \ldots, v_{d+1,k}): \ \forall i \ v_{d+1,i} \in V_d$ and $(v_{d+1,i}, v_{d,1}) \in E.$

The pruning formula is as follows: If $d - 1 + Degree(G_d) \le CBC,$ where CBC is the size of the current best clique, then we prune, since the size of the largest possible clique (formed by expanding any vertex of G_d) would be less than or equal to CBC. If we are at depth 1 and this inequality holds, then we stop; we have found the maximum clique.

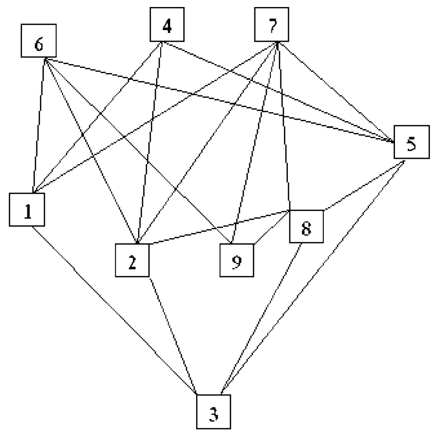
The re-sorting of vertices during vertex-colouring can be used in the *Degree* function calculation to speed up the algorithm. It means that instead of calculating the degree of a subgraph each time at a depth, we will calculate it only the first time and later will just adjust it by the following rule: if the next vertex to be expanded at this depth is in the same colour class as the previous one, then the degree remains the same, otherwise it should be decreased by 1 (there are no more vertices from the previous vertex colour class).

Algorithm for the maximum clique problem

CBC – current best (maximum) clique
d – depth
G_d – subgraph of G formed by vertices existing at depth d

```
Step 0. Heuristic vertex-colouring: Find a vertex-colouring and reorder
        vertices so that the first vertices belong to the last found colour class,
        then come vertices of the last colour class but one, etc. – the last vertices
        should belong to the first colour class. Note: It is advisable to use a
        special array to solve the order of vertices to avoid changing adjacency
        matrix during reordering of vertices.
Step 1. Initialization: $d = 1$.
Step 2. Control: If the current depth may contain a larger clique than already
        found:
        if $d - 1 + Degree(G_d) \le |CBC|,$ then go to step 5.
Step 3. Expand vertex: Get the next vertex to expand. If all vertices have been
        expanded or there are no vertices, then check if the current clique is the
        largest one. If yes, then save it and go to step 5.
Step 4. The next depth: Form a new depth by selecting all remaining vertices
        that are connected to the vertex being expanded at the current depth;
        $d = d + 1;$
        go to step 2.
Step 5. Step back:
        $d = d - 1;$
        delete the expanded vertex from the analysis at this depth;
        if $d = 0,$ then go to end, otherwise go to step 2.
End: Return the maximum clique.
```

Example. Consider the graph shown below and steps of the algorithm shown in Table 1. We determine colour classes one by one in a greedy manner as long as uncoloured vertices exist. This trivial algorithm on finding vertex-colouring gives



an acceptable result on average. The vertices are also re-sorted in an order they are added into colour classes. Thus, vertex-colouring gives the following result:

colour class $1 = \{1,2,5,9\}$,
colour class $2 = \{3,4,6,7\}$,
colour class $3 = \{8\}$;
the order of vertices is the following: $\{8,7,6,4,3,9,5,2,1\}$.

Table 1. Example of algorithm work. *CBC* – current best clique

Depth	Vertices	Algorithm process
Depth 1	8, 7, 6, 4, 3, 9, 5, 2, 1	The grey vertex $v_{d,i}$ ($v_{1,1}$) to be expanded
Depth 2	7, 3, 9, 5, 2	The grey vertex $v_{d,i}$ ($v_{2,1}$) to be expanded
Depth 3	9, 5, 2	The grey vertex cannot be expanded, so *CBC* is {8,7,9}, size = 3
Depth 3	9, 5, 2	We prune, since $d - 1 + Degree = 2 + 1 = 3 \leq 3$ (size of *CBC*). *Degree* = 1, since remaining vertices are 5 and 2. All of them belong to colour class 1. Thus, the number of existing colour classes is 1
Depth 2	7, 3, 9, 5, 2	We prune, since $d - 1 + Degree = 1 + 2 = 3 \leq 3$ (size of *CBC*). *Degree* = 2, since remaining vertices (3,9,5,2) belong to colour classes 1 and 2
Depth 1	8, 7, 6, 4, 3, 9, 5, 2, 1	We prune, since $d - 1 + Degree = 0 + 3 = 3 \leq 3$ (size of *CBC*). *Degree* = 3, since remaining vertices belong to colour classes 1, 2, and 3. Depth is 1, therefore we stop

The maximum clique is {8,7,9}, size = 3.

3. COMPUTATIONAL RESULTS AND DISCUSSION

This section describes the results showing the efficiency of the new algorithm. As we mentioned earlier, a very simple and effective algorithm for the maximum algorithm problem proposed by Carraghan and Pardalos [11] was used as a benchmark in the Second DIMACS Implementation Challenge [12]. Besides, use of that algorithm as a benchmark is advised in one of DIMACS annual reports [13]. Therefore, we compared that algorithm with the new one. Moreover, the algorithm described in the present paper is nothing more than the Carraghan and Pardalos algorithm [11] (we will call it the base algorithm) with the concept of pruning by independent sets (colour classes) introduced. In addition, we have chosen an algorithm proposed by Östergård [14] to be used in the comparison, since it is reported to be faster than the Carraghan and Pardalos algorithm [11]. Moreover, it is another modification of the Carraghan and Pardalos algorithm [11]. The fact that Östergård's algorithm and the new one are just slight modifications of the base algorithm, allows us to assume that worse cases of those algorithms are practically the same as worse cases of the base algorithm. If this is true, then comparison of those algorithms in worse cases cannot give a result different from that obtained by comparison at random graphs. This assumption has been proved practically, on the basis of thousands of experiments we have performed with different random and DIMACS graphs. However, our theoretical proof is postponed to further research, since we believe that random graphs are good enough for an initial try of the new algorithm. This allows us to concentrate initially on random graphs.

Our results are presented as a ratio of time spent by algorithms to find the maximum clique, so the same results can be reproduced on any platform/ computer, etc. The algorithms compared were programmed using the same programming language and the same programming technique (since the new and Östergård [14] algorithms are just modifications of the base algorithm). The greedy algorithm was used to find a vertex-colouring.

We will first look at random graphs. For each entry, 100 graphs were generated and used as an input for each algorithm (Table 2). It is easy to see that both algorithms are faster than the base algorithm, but the new one is certainly faster than the Östergård [14] algorithm. The greatest speed difference is reached on dense graphs, where the new algorithm is 50 times faster than the base algorithm and 23–24 times faster than the Östergård [14] algorithm. The reason for that lies in the fact that the new pruning technique still works at those densities, while the pruning techniques of other algorithms practically do not prune the branches of the maximum clique search tree.

According to the first step of the algorithm, finding an efficient vertex-colouring can be treated as a separate problem. This problem is an NP-hard task; therefore we had to use a heuristic. A heuristic algorithm is an algorithm which

1. does not guarantee the best result, but finds a result that is sufficiently close to the best one;
2. is faster than an exact algorithm. In our case we use a polynomial heuristic – the result is found in a polynomial time.

Table 2. Benchmark results at random graphs

Number of vertices	Edge density, %	*PO*	*New*
1000	0.1	1.0	1.0
800	0.2	1.0	1.2
500	0.3	1.0	1.4
500	0.4	1.1	1.6
300	0.5	1.2	1.8
200	0.6	1.2	2.0
100	0.7	1.4	5.0
100	0.8	1.8	11.0
100	0.9	2.2	50.1

PO – time needed to find the maximum clique by the Carraghan and Pardalos [11] algorithm divided by time needed to find the maximum clique by the Östergård [14] algorithm.
New – time needed to find the maximum clique by the Carraghan and Pardalos [11] algorithm divided by time needed to find the maximum clique by the new algorithm.
For example, 11.0 in the column *New* means that the Carraghan and Pardalos [11] algorithm requires 11-fold more time to find the maximum clique than the new algorithm.

The vertex-colouring step affects the overall result in the following ways:

1. the closer the number of colour classes is to the size of the maximum clique, the faster the maximum clique will be found because of more effective pruning;
2. the more time we spend on vertex-colouring, the slower our algorithm works in general (since the vertex-colouring subroutine is included into the main algorithm and its time should be taken into account).

Moreover, the algorithm can evaluate without changing core steps, by inventing a new and more effective heuristic algorithm for the vertex-colouring. The results obtained by the heuristic algorithm used to find a vertex-colouring applied in the present work are given in Table 3.

Table 3. Number of colour classes by a greedy vertex-colouring

Number of vertices	Edge density, %	Average size of the maximum clique	Number of colour classes	Number of colour classes containing only 1 vertex
100	0.10	3.88	7.16	0.40
100	0.20	5.08	10.36	0.48
100	0.30	6.52	13.88	0.64
100	0.40	8.24	17.20	0.92
100	0.50	10.44	20.76	1.12
100	0.60	13.60	24.80	1.56
100	0.70	18.00	30.00	1.76
100	0.80	24.04	37.24	3.16
100	0.90	34.36	46.08	4.80
100	0.99	69.56	71.20	42.48

It is obvious that effective results of the new algorithm were not reached on the best splitting of vertices into colour classes – the number of colour classes is approximately 25–50% larger than the size of the maximum clique. This difference can be explained by the fact that a quite easy/rough heuristic was used for the vertex-colouring.

4. CONCLUSION

This paper describes a new algorithm for finding the maximum clique on arbitrary undirected graphs. The algorithm was initially designed to be optimal on dense graphs, however, it performs better at all densities than the algorithms published earlier. To show the efficiency of our algorithm, we compared it with an algorithm advised to be used as a benchmark algorithm for finding the maximum clique [12] and with an algorithm that is considered to be the fastest at the moment [14]. The greatest difference in speed of finding the maximum clique is reached on dense graphs. The reason is that the new pruning technique works effectively, while other algorithms degenerate the exhaustive search for the maximum clique at such densities. Moreover, the new algorithm is easy to implement.

Another advantage of the new algorithm is its ability to concatenate an exact and a heuristic algorithm within the scope of one algorithm, making the heuristic algorithm an important part of the exact one (usually heuristics are used to set boundaries and then the maximum clique is searched from scratch; furthermore, as a rule, those boundaries quickly lose topicality). An open problem at the moment is improving the performance of the algorithm by finding a better method for vertex-colouring than the greedy way: to have less colour classes at a reasonably short time (so that the general speed of the algorithm is not decreased).

REFERENCES

1. Garey, M. R. and Johnson, M. R. *Computers and Intractability: a Guide to the Theory of NP-completeness*. Freeman, New-York, 1979.
2. Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cooke, J., Hopkins, M. E. and Markstein, P. Register allocation via colouring. *Comput. Lang.*, 1981, **6**, 47–57.
3. Azar, Y. and Regev, O. On-line bin stretching. *Theor. Comput. Sci.*, 2001, **268**, 17–41.
4. Du, J. and Leung, J. Y.-T. Complexity of scheduling parallel task systems. *SIAM J. Discrete Math*., 1989, **2**, 473–487.
5. Coffman, E. G. Jr., Garey, M. R. and Johnson, D. S. Approximation algorithms for bin packaging: a survey. In *Approximation Algorithms for NP-Hard Problems* (Hochbaum, D. S., ed.). PWS Publishing Company, Boston, 1996, 46–93.
6. Bischof, S. and Mayr, E. W. On-line scheduling of parallel jobs with runtime restrictions. *Theor. Comput. Sci.*, 2001, **268**, 67–90.
7. Peleg, D. Local majorities, coalitions and monopolies in graphs: a review. *Theor. Comput. Sci.*, 2002, **282**, 231–257.

8. Davidson, S. B., Garcia-Molina, H. and Skeen, D. Consistency in portioned networks. *ACM Comput. Surv.*, 1985, **17**, 341–370.
9. Herlihy, M. P. *Replication Methods for Abstract Data Types*. Ph.D. Thesis, Massachusetts Institute of Technology, 1984, MIT/LCS/TR-319.
10. Sinharoy, B. and Szymanski, B. Memory optimization for parallel functional programs. *Comput. Syst. Eng*., 1995, **6**, 415–422.
11. Carraghan, R. and Pardalos, P. M. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.*, 1990, **9**, 375–382.
12. Johnson, D. S. and Trick, M. A. (eds.). *Cliques, Colouring and Satisfiability: Second DIMACS Implementation Challenge*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26. American Mathematical Society, 1996.
13. DIMACS, Center for Discrete Mathematics and Theoretical Computer Science. *Annual Report*, Dec. 1999.
14. Östergård, P. R. J. A fast algorithm for the maximum clique problem. *Discrete Appl. Math.*, 2002, **120**, 197–207.

Optimeerimise probleemid: täpne algoritm suurima kliki leidmiseks suure tihedusega graafidel

Deniss Kumlander

On käsitletud suurima kliki leidmise probleemi rakenduslike algoritmide vaatenurgast. Kõige raskem juhtum selle probleemi lahendamiseks on suure tihedusega graafid. Töös on välja pakutud algoritm, mis leiab suurima kliki suure tihedusega graafidel kuni 50 korda kiiremini kui hetke parimad. Uue algoritmi konstrueerimisel on aluseks võetud hetkel kõige kiirem, Carraghani ja Pardalosi algoritm. Seejuures on arvestatud fakti, et sõltumatu hulga tipud ei või osaleda formeeruvas suurimas klikis. See annab võimaluse hinnata täpsemini formeeruva kliki potentsiaalset suurust ja oluliselt kiiremini otsustada, kas formeeruv klikk saab tulla suurim või mitte. Artiklis on esitatud algoritmi töö tulemused ja arutelud nende üle.