Randomized heuristic for the maximum clique problem

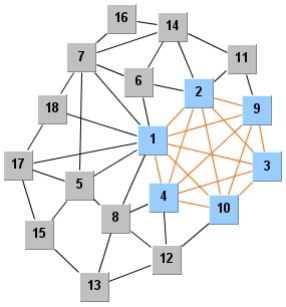
Shalin Shah
sshah100@jhu.edu

Abstract

A clique in a graph is a set of vertices that are all directly connected to each other i.e. a complete sub-graph. A clique of the largest size is called a maximum clique. Finding the maximum clique in a graph is an NP-hard problem and it cannot be solved by an approximation algorithm that returns a solution within a constant factor of the optimum. In this work, we present a simple and very fast randomized algorithm for the maximum clique problem. We also provide Java code of the algorithm in our git repository. Results show that the algorithm is able to find reasonably good solutions to some randomly chosen DIMACS benchmark graphs. Rather than aiming for optimality, we aim to find good solutions very fast.

Keywords:maximum clique, randomized algorithm, fast heuristic, approximation algorithm, k-opt moves, local search, open source, Java, random restart

1 Introduction



The maximum clique problem is one of the harder NP-hard problems in graph theory. It has applications in several domains like computer science, recommender systems and operations research. A clique is a complete sub-graph i.e. all vertices are connected to all other vertices. A maximum clique is a clique of maximum size. A graph may have more than one maximum cliques.

A clique is a lower bound on the chromatic number in a graph coloring problem.

The maximum clique problem is a well studied problem. [1] and [2] are good beginning articles to read about this problem. This work [3] provides a fast algorithm and presents results for several DIMACS graphs. This work [4] is a survey on maximum clique algorithms.

Rather than aiming at finding the maximum clique, this randomized algorithm tries to find large cliques that can be used further in other algorithms. For instance, this algorithm is used in [5] as a first step in an algorithm to color a graph, with really good results. We also present Java code in the following git repository [6]:

`https://github.com/shah314/clique2`

If optimality is important, we refer the reader to the following implementation [7]:

`https://github.com/shah314/clique`

2 Methods

A graph $G(V, E)$ is a set of nodes V and a set of edges E. Each edge in E connects two vertices. A graph can be implemented in an adjacency matrix or an adjacency list format. We use a list format which makes the implementation more amenable to very large graphs.

We first read the DIMACS graph and then sort the vertices in decreasing order of the node degrees. A degree of a node is the number of nodes in its neighborhood. Then, we initialize a clique with the node with the highest degree. Then, in the immediate neighborhood of the first vertex, we complete the clique iteratively by adding vertices in the induced sub-graph of the reach of the nodes already in the clique in decreasing order. Call this initial clique gBest.

Then, we randomly remove 2 nodes from the clique and then greedily complete the clique. If the new clique is larger, update gBest. These can be called 2-opt moves.

Then, for each vertex in the graph, we perform 1-opt moves to improve the clique.

As the results section shows, the algorithm is able to find good solutions to randomly chosen DiMACS instances.

Table 1: Results obtained by running the algorithm [6] on some DIMACS instances

Instance	**Nodes**	**Edges**	**Best Known**	**This Work**
p_hat1500-3	1500	847244	94	93
C2000.5	2000	999836	16	15
p_hat700-1	700	60999	11	9
C500.9	500	112332	57	53
brock800_4	800	207643	26	19
gen400_p0.9_75	400	71920	75	50

3 Results

Table 1 shows the results of running the code on a random sample of DIMACS graphs. As the results show, the Java implementation is able to find reasonably good cliques, very fast, for almost all of the instances. The algorithm can be used to find good cliques for very large graphs as we use an adjacency list format of storing a graph.

4 Conclusion

In this work, we presented a simple fast randomized search heuristic to compute good large cliques in fairly large graphs. It is not the goal of this work to find maximum cliques. Rather, the goal is to find large maximal cliques for use in further applications. For example, [5] uses our algorithm as a first step in an algorithm to find the chromatic number of a graph, as the clique number can be seen as a lower bound on the chromatic number. We also present Java code of our algorithm which is simple and easy to use.

References

[1] Panos M Pardalos and Jue Xue. The maximum clique problem. *Journal of global Optimization*, 4(3):301–328, 1994.

[2] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[3] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1):197 – 207, 2002. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.

[4] Qinghua Wu and Jin-Kao Hao. A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693 – 709, 2015.

[5] Shalin Shah. Jcol: A java package for solving the graph coloring problem. *Journal of Open Source Software*, 5(48):1843, 2020.

[6] https://github.com/shah314/clique2.

[7] Shalin Shah. Gclique: A genetic algorithm for the maximum clique problem in java, 2014.