A GENETIC ALGORITHM FOR DETERMINING THE THICKNESS OF A GRAPH

Erkki Mäkinen, Timo Poranen
and Petri Vuorenmaa

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
INFORMATION SCIENCES
UNIVERSITY OF TAMPERE

REPORT A-2000-5

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
SERIES OF PUBLICATIONS A
A-2000-5, MARCH 2000

A GENETIC ALGORITHM FOR DETERMINING THE THICKNESS OF A GRAPH

Erkki Mäkinen, Timo Poranen
and Petri Vuorenmaa

University of Tampere
Department of Computer and Information Sciences
P.O.Box 607
FIN-33014 University of Tampere, Finland

ISBN 951-44-4793-X
ISSN 1457-2060

A genetic algorithm for determining the thickness of a graph

Erkki Mäkinen[1], Timo Poranen[2] and Petri Vuorenmaa[3]

Dept. of Computer and Information Sciences, P.O. Box 607,
FIN-33014 UNIVERSITY OF TAMPERE, Finland

Abstract

The thickness of a graph is the minimum number of planar subgraphs into which the graph can be decomposed. Determining the thickness of a given graph is known to be a NP-complete problem. This paper discusses the possibility of determining the thickness of a graph by a genetic algorithm. Our tests show that the genetic approach outperforms the earlier heuristic algorithms reported in the literature.

[1]Work supported by the Academy of Finland (Project 35025), e-mail: em@cs.uta.fi
[2]Corresponding author, e-mail: tp@cs.uta.fi
[3]e-mail: pv54521@uta.fi

1 Introduction

One way to characterize the embeddability of a graph G is to determine its thickness $\theta(G)$, i.e., to determine the minimum number of planar subgraphs into which G can be decomposed. The thickness of complete graphs and complete bipartite graphs is known [7], but on the other hand, very little is known about the thickness of an arbitrary graph (see [1, 7, 12] for recently results, and [17] for a survey).

Determining the thickness of a given graph is known to be a NP-complete problem [15]. Hence, heuristic methods are needed for finding the thickness. Cimikowski [6] has introduced certain heuristics with reasonable performance. His heuristics are based on algorithms for finding a maximal planar subgraph of a non-planar graph [5, 13]. In this paper we study the possibilities of applying genetic algorithms for determining the thickness of a given graph. Our tests show that the genetic approach outperforms those presented earlier in the literature.

A well-known approach is to extract maximal planar subgraphs from original graph and to continue this until the remaining graph is planar. This gives as an approximation for the thickness.

To find maximal planar subgraphs there are two classical algorithms. The first one [5] is based on Hopcroft and Tarjan's planarity testing algorithm [10] and the second one [13] is based on Booth and Lueker's [4] PQ-tree approach. Cimikowski has first applied these approaches to determine the thickness of arbitrary graphs in [6]. Another implementation of these approaches can be found in [17]. We call these heuristics (following [6, 17]) $Thick_{HT}$ and $Thick_{PQ}$.

Jünger and Mutzel have also tested a branch and cut algorithm to determine the thickness of a graph [17, 11]. Their algorithm, $Thick_{JM}$, produces only slightly better results than $Thick_{HT}$ or $Thick_{PQ}$.

Extracting a maximal planar subgraph from original graph will not necesserily give the solution for the thickness problem. A counterexample where removing maximal planar subgraph leads to a non-optimal result can be found from [17].

2 Genetic algorithms

The general principle underlying genetic algorithms is that of maintaining a population of possible solutions, which are often called individuals. In our problem a population is a set of partitions of the edges of the input graph. The number of partitions is fixed to be constant in each run of the genetic algorithm. The population undergoes an evolutionary process which imitates the natural biological evolution. In each generation better individuals have greater possibilities to reproduce, while worse individuals have greater possibilities to die and to be replaced by new individuals. To distinguish between "good" and "bad" individuals we need an evaluation function. Our evaluation function is presented in the next section.

The general structure of a genetic algorithm is as follows (see e.g. [16] for further details concerning genetic algorithms):

```
procedure ga
begin
  t := 0;
  create the initial population P(0);
  evaluate the initial population;
  while not Termination-condition do
      t := t + 1;
      select individuals to be reproduced;
      recombine (i.e. apply genetic operations to create the new
          population P(t));
      evaluate(P(t));
  od
end;
```

There are several parameters to be fixed. First, we have to decide how to represent the set of possible solutions. In "pure" genetic algorithms only bit string representations were allowed, but we allow any representation that makes efficient computation possible. In our algorithm we divide edges to partitions, which represents subgraphs of original graph.

Second, we have to choose an initial population. We create initial populations by a method similar to $Thick_{HT}$ heuristic with information on the lower bound for the thickness.

Third, we have to design the genetic operations which alter the composition of children during reproduction. The two basic genetic operations are the *mutation* operation and the *crossover* operation. Mutation is a unary operation which increases the variability of the population by making pointwise changes in the representation of the individuals. Usually crossover combines the features of two parents to form two new individuals by swapping corresponding segments of parents' representations. In our genetic algorithm also crossover operation is unary. It is like a mutation, but it modifies the individual much more than mutation.

3 The algorithm

In this section we introduce our genetic algorithm ($Thick_{GA}$), but first we give some theoretical bounds for thickness. We use these results as lower and upper bounds to direct the genetic algorithm.

Theorem 3.1 *[2] For complete graphs, $\Theta(K_n) = \lfloor \frac{n+7}{6} \rfloor$, except that $\Theta(K_9) = \Theta(K_{10}) = 3$.*

As an example, see Figure 1 where a decomposition of K_9 into three planar subgraphs is shown.

Theorem 3.2 *[3] For complete bipartite graphs, $\Theta(K_{n,n}) = \lfloor \frac{n+5}{4} \rfloor$.*

Theorems 3.1 and 3.2 above can be used as upper bounds for graphs, which are not complete. The next (folklore) theorem is based on Euler's formula. If a graph is planar, it has at most $3|V| - 6$ edges. This formula can be used to derive a lower bound for thickness.

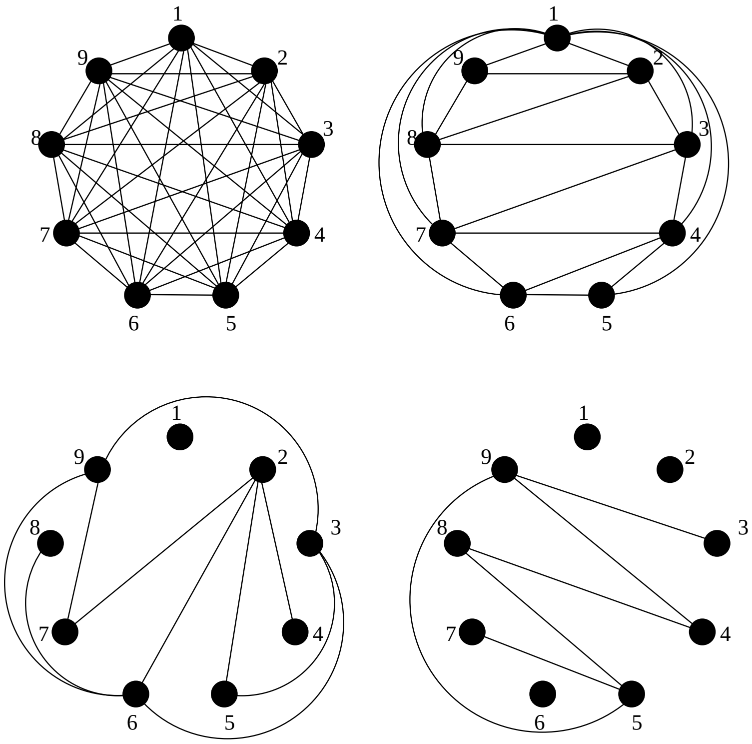

Figure 1: K_9 and a minimum planar decomposition of K_9.

Theorem 3.3 *Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$, then* $\Theta(G) \geq \lceil \frac{m}{3n-6} \rceil$.

In our algorithm every partition of edges is a possible solution, provided that every subset in the partition induces a planar subgraph. The number of such subsets is an upper bound for the thickness of the graph in question.

We obtain an initial result for thickness by using $Thick_{HT}$ heuristics until all subsets are planar. The number of such subsets is an upper bound for the thickness of the given graph. We also know a lower bound (see Theorem 3.3) for arbitrary graphs or exact value for thickness (see Theorem 3.1) for complete graphs and for complete bipartite graphs (see Theorem 3.2).

Let k be the lower bound for the thickness obtained from Theorem 3.1, 3.3 or 3.2. We divide edges into k subsets with maximal planarisation algorithm such that the resulting subsets $P_1, ..., P_{k-1}$ consist of edges from maximal planar subgraphs. The last subset P_k consists of the remaining edges. Clearly, the first $k-1$ subsets are planar and if also the last subset is planar, we have achieved a possible solution. If the last subset is not planar, genetic algorithm tries to make it planar without affecting the planarity of the other subsets.

Genetic algorithm produces new generations until the terminal condition (to be fixed later) is reached. Algorithm also terminates, if a solution is found. If $Thick_{GA}$ is not able to find a solution (to make last subset planar), we increment the number of subsets and start from beginning. In the case where no solution is found before the number of subsets given by $Thick_{HT}$ is reached, we output the result of $Thick_{HT}$.

Let $G = (V, E)$ be a graph. If G is planar, there is only one planar subset of edges, the set E. If $G = K_9$ then one solution is any collection of subsets of the edges which induces planar subgraphs. The optimal solution for K_9 has three subsets (see Figure 1). For arbitrary graphs we don't know whether or not the obtained solution is optimal, we only know lower and upper bounds for thickness.

procedure $Thick_{GA}$
begin
  t := 0;
  p := lower bound for thickness;
  ub := solution from $Thick_{HT}$ algorithm ;
  while $p < ub$ or **not** solution found **do**
    create the initial population P(0) with maximal planarisation algorithm
      such that first $p-1$ subsets are planar and rest edges are in subset p;
    evaluate the initial population;
    while not solution found or terminal condition **do**
      t := t + 1;
      select best individuals to be saved for use in future;
      select individuals to be reproduced;
      apply mutation and crossover operations to individuals;
      create the new population P(t) from this generation and
        from earlier generations;
      evaluate(P(t));
    od
  if solution found **exit**;
  p := p + 1;
  od
end;

Now we are ready to introduce our evaluation function. If a is the number of K_5 subgraphs, b is the number of $K_{3,3}$ subgraphs and c is the number of edges violating planarity in the last subset P_k, then we have following form for evaluation function

$$ef = 3 * a + 2 * b + c.$$

Evaluation function tells how "bad" is the last subset. If evaluation function gets value 0, we have succeeded to make the last subset planar, and we have found a possible solution to the problem. Values a and b are counted by starting to search K_5 subgraphs from vertices, whose degree is greater than or equal to four and $K_{3,3}$ subgraphs from vertices, whose degree is greater than or equal to three. Value c is counted by the maximal planarization algorithm. If an edge does't belong to a maximal planar subgraph, it violates planarity and increments value c. The numbers of K_5 and $K_{3,3}$ subgraphs are approximations. For example, if we find K_6 subgraph, it will increase value a only by two.

To define the mutation operation, consider a graph with 10 edges which are divided into three subsets $P_1 = \{e_1, e_2, e_6, e_9\}$, $P_2 = \{e_3, e_4, e_5, e_7\}$ and

$P_3 = \{e_8, e_{10}\}$. We denote the number of edges by k and the number of subsets by p. First we generate a random real number from the interval [0..1]. If the random number is not greater than the mutation rate (to be fixed later) we perform a mutation operation as follows. We generate one random integer i from [1..k]. Integer i tells the edge to be moved to last subset p. Suppose that we have $i = 3$. Then we move edge e_3 from subset P_2 to subset P_3. Now subsets $P_1, \ldots, P_{p-1}$ are still planar.

Usually the crossover operation transforms two individuals into two new individuals, but in our algorithm crossover is also a unary operation. Also now we generate first a random real number from [0..1]. Crossover operation will be applied to an individual if generated random number is not greater than the crossover rate (to be fixed later). If the last subset has K_5 or $K_{3,3}$ as subgraphs, as many as possible of their edges will be moved to other subsets (without affecting the planarity of them). If we cannot easily break K_5 or $K_{3,3}$ we search maximal planar subgraphs from the last subset. Then we also find edges which violates planarity. First we try to move those edges to other subsets. If they cannot be moved, we try to swap them with edges in the other subsets.

Clearly, if crossover or mutation operation succeeds to make the last subset planar, we have found a possible solution for the thickness problem. In case that all individuals have edges violating planarity in the last subset, the new population will consist of best individuals from this generation (with respect to our evaluation function) and best individuals from earlier populations. Genetic algorithm saves few best individuals from each generations. An individual is better than another one if it has lower value for evaluation function.

4 Tests

In this section we describe our software and test runs. The implementation uses the GAlib [8] genetic algorithm package, and for data structures and graph algorithms we used LEDA [14].

Population size is an important parameter of a genetic algorithm. Population size should be large enough to give an unbiased view of the search space. On the other hand, too large population size makes the algorithm too slow. We tested the effect of population size by counting the number generations needed to find a solution for thickness. If the size of population was more than 50, results was not significantly different than with the population size 50. Only the time of computationtime increased remarkably.

The intutition behind the mutations is that they increase the variability of population. Naturally, there is again a trade-off situation: if mutation rate becomes too large, the algorithm wanders aimlessly in the search space. In our algorithm also the crossover operations are unary, and the idea behind them was to make last subset planar. We made test runs with different values of mutation rate and crossover probability and found out that solution was found fastest with mutation rate 0.2 and crossover probability 0.9.

The number of generations is also an important parameter of genetic algorithms. If the situation is stabilized, there is no sense to create new generations. In our test runs, we noticed that the minimum value for evaluation function was found average in 500 generation. Often the result for thickness was obtained in first ten generations, but there were few test runs, where making the last subset

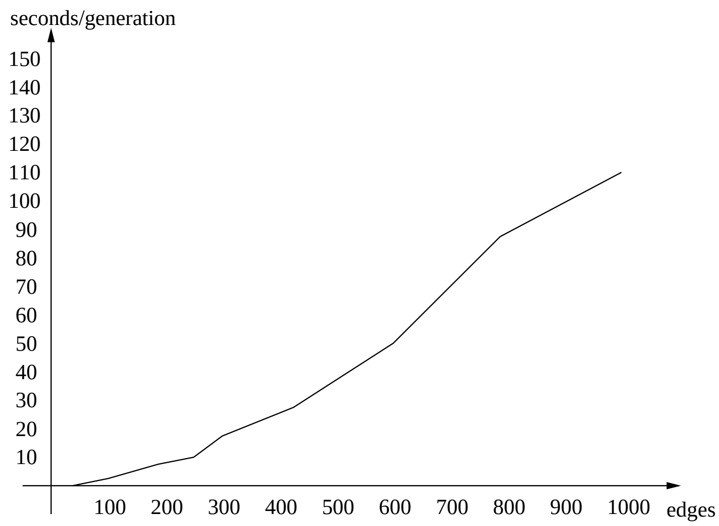

Figure 2: Evaluation time in seconds for one generation in function of edges.

planar needed over 1000 generations.

We have compared our results to four other heuristics, $Thick_{HT}$, $Thick_{PQ}$, $Thick2$ and $Thick_{JM}$, for determining the thickness. $Thick2$ heuristic [6] is a combination of $Thick_{HT}$ and $Thick_{PQ}$. $Thick2$ uses also local optimization strategy for maximal planar subgraphs. Figure 3 shows results for the thickness of complete graphs K_n where $n = 10, 15, \ldots, 50$. Values with brackets are taken from [6] and all other are taken from [17]. If there was different value for same graph in [6] and [17], we have taken them both. The results where $Thick_{GA}$ has improved earlier results, are marked with $*$.

Genetic algorithm has outperformed earlier heuristic results with $n < 30$, but with higher values of n there are no improvements.

In Figure 4 there are results for complete bipartitite graphs. Now every result for $Thick_{GA}$ has same value than best earlier heuristic.

In Figure 5 there are results for random graphs with same number of vertices and edges as in [6]. Graphs were generated with LEDA's [14] functions for random graphs.

5 Conclusions

We have found out that genetic algorithm can be used to sharpen the results obtained by previously known heuristics for the thickness of the graph. Best results are achieved for complete graphs with less than 30 vertices.

n	$Thick_{HT}$	$Thick_{PQ}$	$Thick2$	$Thick_{JM}$	$Thick_{GA}$	$\Theta(K_n)$
10	3	3	(3)	3	3	3
15	4	4	(4)	4	3*	3
20	5	6(5)	(5)	5	4*	4
25	7	7(6)	(6)	6	5*	5
30	8(9)	8	(7)	7	7	6
35	9(10)	9	(8)	8	8	7
40	10(11)	11(10)	(9)	9	9	7
45	11(12)	12(11)	(10)	10	10	8
50	13(15)	13(14)	(11)	11	11	9

Figure 3: Heuristic results for complete graphs K_n.

n	$Thick_{HT}$	$Thick_{PQ}$	$Thick2$	$Thick_{JM}$	$Thick_{GA}$	$\Theta(K_n)$
10	4	4	(3)	4	3	3
15	6	6	(5)	5	5	5
20	7(8)	7	(6)	7	6	6
25	9(10)	9	(7)	8	7	7
30	10(12)	10(11)	(9)	9	9	8
35	12(14)	12	(10)	11	10	10
40	13(16)	14(15)	(13)	12	12	11
45	15(17)	15(16)	(13)	13	13	12
50	16(19)	17(18)	(15)	14	15	13

Figure 4: Heuristic results for complete bipartitite graphs $K_{n,n}$.

References

[1] I. Aho, E. Mäkinen, and T. Systä, "Remarks on the thickness of a graph," *Info. Sci.*, 108 (1998), 1–4.

[2] V.B. Alekseev, and V.S. Gonchakov, "Thickness for arbitrary complete graphs," *Mat. Sbornik.*, 143 (1976), 212–230.

[3] L.W. Beineke, F. Harary, and J.W. Moon, "On the thickness of the complete bipartite graphs," *Proc. Cambridge Phil. Soc.*, 60 (1964), 1–5.

[4] K.S. Booth, and G.S. Lueker, "Testing for the consecutive ones property, Interval graphs and graph planarity testing using PQ-tree algorithms," *J. Comp. and System Sci.*, 13 (1976), 335–379.

[5] J. Cai, and X. Han, and R.E. Tarjan, "An $O(m \log n)$-time algorithm for maximal planar subgraph problem" *SIAM J. Comput.*, 22 (1993), 1142–1162.

[6] R. Cimikowski, "On heuristics for determining the thickness of a graph," *Info. Sci.*, 85 (1995), 87–98.

[7] A.M. Dean, J.P. Hutchinson, and E.R. Scheinerman, "On the thickness and arboricity of a graph," *J. Comb. Theory (B)*, 52 (1991), 147–151.

n	m	LB	$Thick_{HT}$	$Thick_{PQ}$	$Thick2$	$Thick_{GA}$	UB
10	25	2	(2)	(2)	(2)	2	2
20	92	2	(3)	(3)	(3)	3	4
30	230	3	(5)	(5)	(4)	4	6
40	311	3	(5)	(5)	(5)	4*	7
50	495	4	(7)	(6)	(5)	5	9
60	556	4	(7)	(6)	(5)	5	11
70	766	4	(8)	(7)	(6)	6	12
80	939	5	(8)	(8)	(7)	7	14
90	1263	5	(10)	(9)	(7)	8	16
100	1508	6	(12)	(11)	(8)	9	17

Figure 5: Heuristic results for random graphs with n vertices and m edges.

[8] galib, software available in http://lancet.mit.edu/ga/.

[9] J.H. Halton, "On the thickness of graphs of given degree," *Info. Sci.*, 54 (1991), 219–238.

[10] J. Hopcroft, and R.E. Tarjan, "Efficient planarity testing," *J. ACM*, 21 (1974), 549-568.

[11] M.Jünger, and P. Mutzel, "Maximum planar subgraph and nice embeddings: Practical layout tools," *Algorithmica*, 16 (1996), 33–59.

[12] M.Jünger, P. Mutzel, T. Odenthal, and M. Scharbrodt, "The thickness of a minor-excluded class of graphs," *Discrete Math.*, 182 (1998), 169–176.

[13] G. Kant, "An $O(n^2)$ maximal planarization algorithm based on PQ-trees," Utrecht University, *Technical Report RUU-CS-92-03*, (1992).

[14] LEDA, software available in http://www.mpi-sb.mpg.de/LEDA/.

[15] A. Mansfield, "Determining the thickness of graphs is NP-hard," *Math. Proc. Camb. Phil. Soc.*, 93 (1983), 9–23.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

[17] P. Mutzel, T. Odenthal, and M. Scharbrodt, "The thickness of graphs: a survey," *Graphs and Combinatorics*, 14 (1998), 59–73.