Applications of Neural Networks to Telecommunications 2
pp. 49–56 (1995). ISBN 0-8058-2084-1

Graph Coloring Algorithms for Assignment Problems in Radio Networks

Francesc Comellas and Javier Ozón

Departament de Matemàtica Aplicada i Telemàtica
E.T.S.E. Telecomunicació, Universitat Politècnica de Catalunya
Campus Nord C-3, Gran Capitán s/n, Barcelona, Catalonia, Spain
comellas@mat.upc.es ozon@mat.upc.es

Abstract

Assignment problems in radio networks, like the channel assignment, may be solved by graph coloring algorithms. In this paper we compare the performance of several recent techniques on a simple coloring problem: the search for a bipartite subgraph with the maximum number of edges of a given graph. The algorithms considered are a neural network, a genetic algorithm, simulated annealing and two heuristics. The results show that one heuristic, a new proposed multiagent system, and the standard simulated annealing technique are faster outperforming the algorithms in the literature. We also show how the algorithms may be easily adapted to deal with the k-coloring of a graph and can therefore be used for solving assignment problems in telecommunications.

1 Introduction

Several issues related to the design of radio networks, as well as other problems in telecommunications, may be formulated as graph coloring problems. Consider, for example, the construction of a packet radio network which uses time division multiplexing. Since a user cannot transmit and/or receive more than one packet at a time, time slots must be associated to channels (one channel for each pair of users) in such a way that all channels

that correspond to a particular user get different time slots. If this problem is modeled by graphs, users correspond to vertices, channels to edges and assigning time slots corresponds to coloring the edges so that incident edges have different colors. In a similar way we may deal with the task of assigning channels to the radio base stations in a spectrum efficient way. Particular instances of these problems are very often NP-complete optimization problems [6] and finding an optimal solution is a computationally hard task. In these cases, heuristic methods and general combinatorial optimization methods are used to obtain an acceptable answer in a reasonable time, see [5, 9, 11].

In this paper we have considered the instance of the coloring problem when only two colors are involved, and we have compared the methods described in the literature (a heuristic search [8] and a neural network [11]) with the techniques introduced in this work (simulated annealing, a genetic algorithm and a multiagent system). In Section 2 we give a short introduction to the terminology used and the exact description of the problem. In Section 3 we describe the algorithms considered. Finally, in Section 4, we present the details of the implementations, the results obtained and the extension of the algorithms to the general case.

2 Graph coloring

A *proper coloring* of a graph $G = (V, E)$ is a function from the vertices of the graph to a set C of *colors* such that any two adjacent vertices have different colors. If $|C| = k$, we say that G is *k-colored*. The minimum possible number of colors for which a proper coloring of G exists is called the *chromatic number* $\chi(G)$ of G. If $\chi(G) = 2$ then G is bipartite.

The problem of finding the chromatic number and a proper coloring of a graph is of great interest for its widespread applications in areas such as scheduling and timetabling and particularly in telecommunications. As many other problems in graph theory, it is NP-complete. In this work a variation of the graph coloring problem, the *bipartite subgraph problem*, has actually been considered. This particular problem, which is also NP-complete, consists of removing the minimum number of edges of a given graph in such a way that the remaining graph is a bipartite graph. Efficient algorithms for this problem are known only for particular graphs, e.g. triangle-free graphs with maximum degree three [2]. A heuristic algorithm, introduced by Hsu in [8], can be applied to any graph and more recently Lee, Funabiki and Takefuji [11] proposed a neural network model for this problem.

On the other hand, simulated annealing and genetic algorithms have outperformed neural network approaches when applied to several NP-complete

graph problems, see [3, 4]. This fact encouraged the authors to look for the application of these techniques to the coloring problem, object of this paper.

3 Combinatorial optimization algorithms

The first technique that we have applied to the bipartite subgraph problem belongs to a general class of algorithms, known as genetic algorithms, which are inspired on the mechanics of natural selection and genetics.

In a genetic algorithm, see Goldberg [7], the starting point is a collection of possible solutions generated at random, known as *population*. A suitable encoding of each solution in the population is used to compute its *fitness* through a *cost function*. At each iteration a new population, or *generation*, is obtained by *mating* the best of the old solutions with one another. To create the next generation, new solutions are formed through *reproduction*, *crossover* and *mutation*. The solutions that will be considered for crossover are probabilistically selected according to the fitness values from the set that constitutes the current generation. This new population becomes the *parent pool*. In the Goldberg's approach a constant number of solutions are selected so that a fixed size population is maintained. Crossover creates two new *child* solutions from two solutions sampled from the parent pool. In this way, fitter parents have a better chance of producing children. This is done for the whole population. Children solutions are obtained by interchanging random parts of their parents. Some randomness is also introduced through the mechanism called *mutation* to ensure that the algorithms avoid getting stuck at local minima. Mutation changes selected parts of a solution. The crossover and mutation operations are done with fixed probabilities, thus ensuring that some solutions from the current generation will be kept in the new generation.

Once a new generation is created, the fitness of all solutions is evaluated and the best solution is recorded. The process is repeated until either the results stabilize or the optimal solution, when it can be identified, is reached.

The main aspects to decide are the representation of the solutions, the cost function and the crossover and mutation operators. Important parameters are the population size and the probabilities of crossover and mutation.

The second algorithm considered uses the simulated annealing (SA) technique which comes from the analogy made between the states of a physical system, e.g. a liquid, and the configurations of a system in a combinatorial optimization problem. If the temperature of the interacting molecules in a liquid is suddenly reduced below its freezing point, the result is a disordered glassy state with an energy higher than the true crystalline ground state. In fact the molecules are in a local energy minimum. On the other hand,

if the temperature of the liquid is reduced slowly (annealing), waiting for equilibrium to be reached before a new reduction is made, the liquid freezes to the solid state through a cooling process that leads to the crystalline state, which is the global energy minimum. In the analogy with the combinatorial optimization problem the parameters being varied are equated with atomic positions in the liquid and its energy is identified with the cost function being optimized. The temperature is then defined as a control parameter related to the probability of accepting changes that worsen the state of the system, thus ensuring a more comprehensive search of the state space. Hence, in the simulated annealing, a change of state that decreases the energy is always accepted, whereas if energy increases, the change is accepted with a certain probability $e^{-\Delta E/T}$, where T is the temperature of the system. At a given temperature, a number of attempts large enough to obtain a good statistical set of trials is performed and thereafter the temperature decreased. This process is repeated and the system is gradually cooled until it is stopped according to some criteria such as a small number of changes accepted and/or a non significant reduction of the energy.

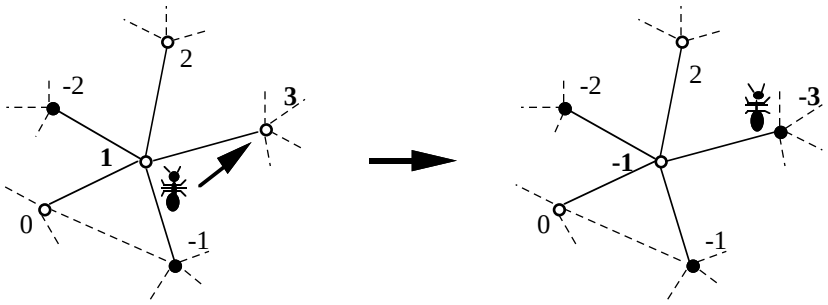

Figure 1: An ant acting on a graph.

The third algorithm we have considered is `ants`, a multiagent system where several autonomous agents work together to obtain good solutions of the bipartite subgraph problem. The `ants` algorithm starts by generating an arbitrary random coloring of the graph which is, in general, non proper, and by randomly allocating a set of agents, called *ants*, on different vertices of the graph. Once it has been initializated, the process begins to evolve by assigning to every vertex a certain value, calculated as the difference between the number of adjacent vertices with the same color and the number of adjacent vertices with different color. After that, every ant moves to the adjacent vertex that has maximum value, changes the color of this vertex and recalculates the values for it and its adjacent vertices (see Figure 1), all that with a certain probability to avoid local minima. Once each ant

has moved, the process is repeated until the algorithm converges to a near-optimal solution.

4 Results and conclusions

As an illustration of the performance of the three proposed algorithms note that we may obtain from the graph with 30 vertices and 50 edges considered by Lee *et al.* [11] a bipartite subgraph with 43 edges, whereas Hsu's heuristic algorithm [8] found only a subgraph with 38 edges and Lee *et al.* with a neural network found a subgraph with 42 edges.

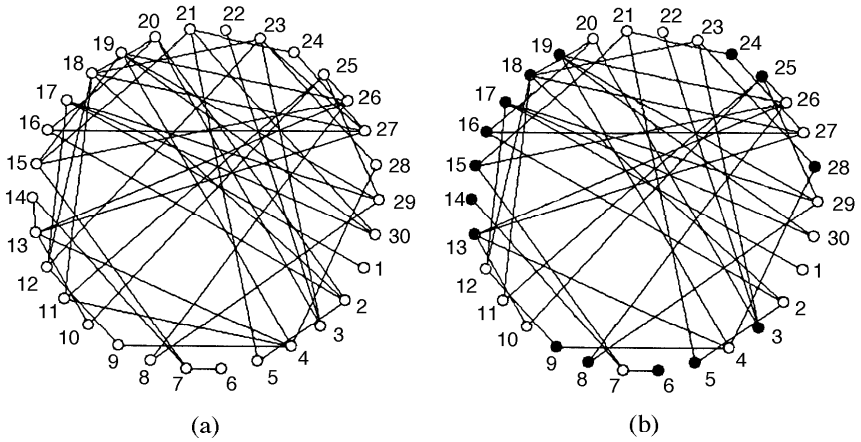

Figure 2: The graph proposed by Lee *et al.* and the solution with 43 edges found in this paper.

For our study, a large number of instances have been simulated to test the three algorithms with random graphs of orders ranging from 10 to 250 vertices and densities of 5%, 15% and 20% (the density of a graph is the ratio between the number of edges that actually has the graph and the maximum number that may contain). For each case 20 simulation runs were performed and compared with the results given by Lee *et al.* in their paper [11]. All simulations were programmed in C (less than 500 lines) and executed on a HP Apollo 715/75.

Possible solutions has been coded as lists where each position represents a vertex of the graph and has values 0 or 1 according to the color assigned to it. The cost function calculates the number of edges that must be removed from the associated graph to have a proper coloring.

In the genetic algorithm, we have used the same crossover and mutation

operators than Goldberg [7]. Figure 3 shows an example of their application. Values for the relevant parameters were taken as follows: a population size of 400 individuals and probabilities of crossing and mutation of 0.95 and 0.01.

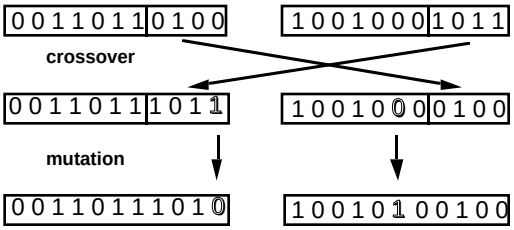

Figure 3: Crossover and mutation operators in the genetic algorithm.

For simulated annealing we have used a standard implementation as described in [1] and [10]. Simulated annealing requires careful tuning of its control parameters to achieve good results. Typical values considered were: initial temperature, T_0, ranging from 0.5 to 1.5, number of iterations for a given temperature $N_k = \alpha|E|$, where $|E|$ is the number of edges of the graph and α is a factor ranging from 1 to 10, and cooling rate exponential, $T_k = 0.9T_{k-1}$.

The third method, our multiagent system, used up to 10 agents (or *ants*) and a probability of 0.9. Simulations show that the total number of agents is not critical for the solution when the number of vertices of the graph is relatively small (less than 1000).

Table 1 shows the results corresponding to graphs with a density of 15%. Similar performances were obtained with different densities.

Finally, the new algorithms presented in this paper may be very easily adapted to solve the problem of finding a k-coloring of a given graph. For the genetic algorithm and the simulated annealing, all that is needed is to change the set of values associated to each vertex from $\{0, 1\}$ to $\{0, 1, \ldots, k-1\}$ in the list that codes a solution. The multiagent system may be also modified to deal with more colors, but in that case the cost function needs to be redefined. We have tested with success the modifications and generated easily a 3-coloring of the Lee graph that uses all its edges.

As a conclusion, the results show that the multiagent system, the simulated annealing and the genetic algorithm presented in this work perform better that the neural network approach. All three algorithms are simple and easy to implement and may be adapted to deal with the general k-coloring problem. Moreover, the multiagent system, besides finding better solutions, runs around ten times faster than simulated annealing and fifty

Node Size	**Hsu**	**NN**	**GA** max.	**GA** mean	**SA** max.	**SA** mean	**ants** max.	**ants** mean
10	6	6	6	6.0	6	5.8	6	5.8
20	24	25	24	22.5	26	24.2	26	24.7
30	49	50	54	52.1	58	52.9	58	53.4
40	90	90	94	89.9	94	91.0	96	91.1
50	128	135	139	135.4	140	137.9	142	137.4
60	191	195	199	191.1	198	193.5	200	193.7
70	246	254	257	251.9	261	257.6	265	258.4
80	311	330	330	326.2	338	331.4	339	331.6
90	390	405	406	402.9	419	412.4	421	413.1
100	478	494	492	485.0	508	502.8	514	503.4
150	1032	1060	1024	1015.0	1089	1080.7	1092	1084.8
180	1476	1502	1437	1420.5	1541	1527.9	1547	1532.2
200	1824	1860	1850	1812.1	1881	1867.4	1883	1870.1
250	2817	2854	2832	2807.6	2881	2866.3	2884	2865.1

Table 1: Comparison between Hsu's algorithm, neural networks, genetic algorithm (GA), simulated annealing (SA) and multiagent algorithm (`ants`) on graphs with density 15%.

times faster than the genetic algorithm. It may be also directly implemented on a parallel computer, thus improving its performance.

Acknowledgment

This work was supported by the *CICYT*, Spain, under grants TIC-92-1228-E, TIC94-0592 and the EU-HCM program ERBCHRX-CT920049.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*, Chichester: John Wiley and Sons, 1989, ISBN 0-471-92146-7.

[2] J.A. Bondy and S.C. Locke. "Largest bipartite subgraph in triangle-free graphs with maximum degree three", *J. Graph Theory*, vol. 10, pp. 477–504, 1986.

[3] F. Comellas. "Using genetic algorithms for planarization problems", Computational and Applied Mathematics I, Eds. C. Brezinski and U. Kulish, Elsevier Science Publishers B.V. (North Holland), pp. 93-100, 1992, ISBN 0-444-89701-1.

[4] F. Comellas and E. Pallarès. "Optimització combinatòria i disseny de xarxes d'interconnexió", *Butll. Soc. Cat. Cièn.*, vol. XIV (2), 1994, pp. 221–234.

[5] M. Duque-Antón, D. Kunz, and B. Rüber. "Channel assignment for cellular radio using simulated annealing", *IEEE Trans. Vehicular Technology*, vol. 42, pp. 14–21, 1993.

[6] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: W.H. Freeman, 1979, ISBN 0-7167-1044-7.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989, ISBN 0-201-15767-5.

[8] C.P. Hsu, "Minimum-via topological routing", *IEEE Trans. Computer-Aided Design*, vol. 2, pp. 235–246, 1983.

[9] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon. "Optimization by Simulated Annealing: an Experimental Evaluation; Part II, Graph Colouring and Number Partitioning", *Operations Research*, vol. 39, pp. 378–406, 1991.

[10] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing". *Science*, vol. 220, pp. 671–680, 1983.

[11] K. C. Lee, N. Funabiki and Y. Takefuji. "A Parallel Improvement Algorithm for the Bipartite Subgraph Problem", *IEEE Trans. Neural Networks*, vol. 3, pp. 139–145, 1992.