A New Genetic Approach For Maximum Independet Set Problem

O. Ugurlu, M.E. Berberler, U. Nuriyev

In this study, The Maximum Independent Set Problem (MIS) is studied and a genetic algorithm to solve the MIS is proposed. In order to scan the solution space more efficiently, the initial population is generated by means of a simple heuristic algorithm. Moreover, a new optimization technique is added to genetic algorithm to enhance the population diversity. The algorithm is implemented in C and is tested on DIMACS benchmark graphs. It is seen that the algorithm can yields optimal solution for most of the instances.

Keywords: evolutionary algorithms, maximum independent set.

ACM 2012: Mathematics of computing → Discrete mathematics → Combinatorics; Mathematics of computing → Discrete mathematics → Graph theory.

UDC: 519.1

Introduction

The maximum independent set problem is the classic one in computer science and graph theory and is known to be NP-Hard [1]. Consider an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges in G. An independent set is a set of vertices of G so that no two vertices of the set are adjacent. The maximum independent set problem calls for finding the independent set of maximum cardinality.

The opposite concept of an independent set is a clique. That is, a clique of G is a subgraph induced by a set S of vertices, which is complete [2]. The maximum clique problem is to find the largest clique in G.

When solving the maximum independent set problem, we obtain solutions for the maximum clique problem and another important graph problem: the minimum vertex cover problem, as well [1].

The maximum independent set problem has many important applications, including combinatorial auctions, graph coloring, coding theory, geometric tiling, fault diagnosis, pattern recognition, scheduling, computer vision, molecular biology, and more recently its application in bioinformatics has become important [3].

Due to the theoretical importance of MIS, a large number of exact, approximate, heuristic and metaheuristic algorithms have been proposed for the maximum independent set problem. Especially, genetic algorithms have been frequently used for solving this problem. Some of these algorithms may be found in [2, 4, 5] and [6].

In this paper, a new genetic algorithm has been considered to find the maximum independent set of the graph and the algorithm has been implemented in C, then it has been tested on DIMACS benchmark graphs. The experimental results show that the proposed algorithm yields good solutions in reasonable times.

The paper is organized as follows: Section II outlines genetic algorithms. Section III describes the proposed genetic algorithm. In section VI, computational efficiency of the proposed algorithm is tested on the DIMACS benchmark instances. Section V summarizes and concludes the paper.

Overview of Genetic Algorithm

The genetic algorithm is a technique, which is based on the natural evolution, for randomized search and optimization. The genetic algorithms have been applied in wide range of studies in solving optimization problems [7].

In a genetic algorithm, a population of candidate solutions to an optimization problem is evolved toward better solutions. The evolution usually starts from a population of randomly generated individuals. In each generation, the fitness of every individual in the population is evaluated, the more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new population [7].

Although GA is probabilistic, in most cases, it produces better population compared to their parent population, because selected parents are the fittest among the whole population set, and the worse individuals die off in successive generations. This procedure is continued until some user-defined termination criteria are satisfied.

$Initialize(Population)$
$Generation \leftarrow 0$
while $Termination\_Criterions\_are\_not\_Satified$ **do**
  $Generation \leftarrow Generation + 1$
  $Selection(Population)$
  $Crossover(Population)$
  $Mutate(Population)$
end

Algorithm 1: The Basic Steps of a Genetic Algorithm

In the next section, we describe the new genetic algorithm for the Maximum Independent Set problem.

A New Genetic Algorithm for MIS Problem

Unlike the classical genetic algorithm, the initial population is not generated randomly in the proposed algorithm. We use the following simple heuristic algorithm for generating initial population.

Heuristic for Initial Population

The algorithm starts by selecting one initial vertex, tries to add its neighbors to independent set. If the neighbor vertex does not make the solution unfeasible, the vertex is added to the solution. After updating the adjacency matrix of the

graph by deleting all edges which are adjacent to the vertex in the solution and its neighbor, algorithm finds the vertex which has the maximum degree and tries to add its neighbor to solution in a similar way. This procedure continues until degrees of all vertices become zero.

The Proposed Genetic Algorithm

We set size of the population to be $\mu \leq 10$ and only 1 generation, because the proposed algorithm converges to global solution easily by means of the heuristic which is used for generating initial population and optimization technique.

To reproduction, each individual of the population is crossed (one point crossover) with all other individuals. Since we work with a small size population, it can be done in reasonable time.

After crossover operation, all individuals are checked for feasibility and if there is an addible vertex that does not make the solution unfeasible when added to it, the algorithm adds the vertex to the solution. Since we have used the following optimization technique, we have not used mutation.

The Optimization Technique

After the crossover operation, the algorithm looks for the addible vertex. However it is almost impossible to find an addible vertex when the solutions converge towards the local optima. To avoid local optima and find addible vertices, we have used the following procedure;

Procedure : If a vertex n is not in a solution and its only one neighbour m is in the solution, then remove m from the solution and add n to the solution.

The *Procedure* provides the population diversity by making small moves around the neighbors of the solution and help to find addible vertices. The vertices used in the Procedure are selected randomly and the *Procedure* is applied to all individuals. The number of applications of the *Procedure* is limited by the number of the vertices in the instance.

Computational Experiments

This section presents the results of computational experiments for the proposed genetic algorithm. All procedures of the algorithm have been coded in C++ language. The experiments have been carried out with an Intel Pentium Core2 Duo 2.6 GHz CPU and 2GB of RAM. The algorithm has been tested on the complement graphs of DIMACS benchmark instances [8]. These graphs are designed in terms of finding maximum cliques, so we have considered the clique benchmark graphs as $\overline{G}$.

The effectiveness of the proposed genetic algorithm has been evaluated using 68 instances. For each of these problems, a total of 10 runs of the algorithm are performed. The results are summarized in Table 1 and 2 for the best results that were encountered during the 10 runs for each instance.

In Table 1 and 2, the first three columns show the type of the instances such as name, cardinality and density of the instances; the fourth one gives the optimum

value, the fifth column denotes the solution value achieved by the proposed algorithm and the last column shows running time of the proposed algorithm.

Table 1. Simulation Results For Dimacs Benchmark Graphs

$\overline{G}$	$\|N\|$	Density	$\alpha(G)$	GA	Time(s)
brock200_1	200	0.745	21	21	0.32
brock200_2	200	0.496	12	11	0.43
brock200_3	200	0.605	15	14	0.36
brock200_4	200	0.658	17	16	0.36
brock400_1	400	0.748	27	24	1.76
brock400_2	400	0.749	29	25	1.96
brock400_3	400	0.748	31	25	1.68
brock400_4	400	0.749	33	25	1.57
brock800_1	800	0.649	23	20	15.18
brock800_2	800	0.651	24	20	15.75
brock800_3	800	0.649	25	20	14.53
brock800_4	800	0.651	26	20	20.059
C125.9	125	0.898	34	34	0.09
C250.9	250	0.899	44	44	0.53
C500.9	500	0.9	57	56	2.98
C1000.9	1000	0.901	68	65	23.60
c-fat200-1	200	0.077	12	12	0.03
c-fat200-2	200	0.163	24	24	0.06
c-fat200-5	200	0.426	58	58	0.04
c-fat500-1	500	0.036	14	14	0.29
c-fat500-2	500	0.073	26	26	0.31
c-fat500-5	500	0.186	64	64	0.53
c-fat500-10	500	0.374	126	126	0.42
DSJC500.5	500	0.5	13	13	3.00
DSJC1000.5	1000	0.5	15	14	36.11
gen200_p0.9_44	200	0.9	44	44	0.31
gen200_p0.9_55	200	0.9	55	55	0.28
gen400_p0.9_65	400	0.9	65	65	1.42
gen400_p0.9_75	400	0.9	75	75	0.95
hamming6-2	64	0.905	32	32	0.01
hamming6-4	64	0.349	4	4	0.01
hamming8-2	256	0.969	128	128	1.34
hamming8-4	256	0.639	16	16	0.28
hamming10-2	1024	0.99	512	512	22.78
hamming10-4	1024	0.829	40	40	3.54
johnson8-2-4	28	0.556	4	4	0.00
johnson8-4-4	70	0,226	14	14	0.01
johnson16-2-4	120	0,235	8	8	0.03
johnson32-2-4	496	0,121	16	16	2.48

As one can see, the proposed genetic algorithm yields optimum solutions for most of the instances. The algorithm has performed better on c-fat, DSJC, gen, hamming, Johnson, p_hat, san, sanr instance families than brock and C instance families and has found optimal solutions for all instances of gen, hamming, Johnson, sanr families. Since we are dealing with an NP-hard problem, it is expected that the

Table 2. Simulation Results For Dimacs Benchmark Graphs

$\overline{G}$	$\|N\|$	Density	$\alpha(G)$	GA	Time(s)
p_hat300-1	300	0.244	8	8	0.09
p_hat300-2	300	0.489	25	25	1.37
p_hat300-3	300	0.744	36	36	0.92
p_hat500-1	500	0.253	9	9	2.75
p_hat500-2	500	0.505	36	36	5.84
p_hat500-3	500	0.752	50	50	4.09
p_hat700-1	700	0.249	11	11	15.09
p_hat700-2	700	0.498	44	44	16.32
p_hat700-3	700	0.748	62	62	9.69
p_hat1000-1	1000	0.245	10	10	19.31
p_hat1000-2	1000	0.49	46	46	23.70
p_hat1000-3	1000	0.744	66	66	14.17
p_hat1500-1	1500	0.253	12	11	21.28
p_hat1500-2	1500	0.506	65	65	27.51
p_hat1500-3	1500	0.754	94	94	16,70
san200_0.7_1	200	0.7	30	30	0.85
san200_0.7_2	200	0.7	18	18	3.31
san200_0.9_1	200	0.9	70	70	0.31
san200_0.9_2	200	0.9	60	60	0.23
san200_0.9_3	200	0.9	44	44	0.29
san400_0.7_1	400	0.7	40	22	4.84
san400_0.7_2	400	0.7	30	30	3.75
san400_0.7_3	400	0.7	22	17	2.75
san400_0.9_1	400	0.9	100	100	2.32
san1000	1000	0.502	15	10	33.98
sanr200_0.7	200	0.697	18	18	0.28
sanr200_0.9	200	0.898	42	42	0.28
sanr400_0.5	400	0.501	13	13	3.57
sanr400_0.7	400	0.7	21	21	2.15
sanr200_0.7	200	0.697	18	18	0.28
sanr200_0.9	200	0.898	42	42	0.28
sanr400_0.5	400	0.501	13	13	3.57
sanr400_0.7	400	0.7	21	21	2.15

performance of algorithms can change as much as the structure of different instances.

Conclusion

Maximum independent set problem is a classic graph optimization problem that is NP-complete even to approximate well. In this paper, a new genetic algorithm for maximum independent set problem has been proposed. We have used a new optimization technique and a simple heuristic for generating initial population in order to improve the performance of the algorithm. The algorithm has been implemented in C language. The performance of the algorithm has been tested on DIMACS clique instances. The experimental results have shown that the algorithm has found optimal solution for all instances of some families and yielded good solutions for all instances.

In future works, it is aimed to decrease the running time of the algorithm and improve the solution quality for some certain instance types.

Acknowledgement

O. Ugurlu gratefully acknowledges support of TUBITAK 2211 program.

References

[1] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[2] X. Liu, A. Sakamoto, and T. Shimamoto, "A genetic algorithm for maximum independent set problems," in *Proceedings of the "Systems, Man, and Cybernetics, 1996., IEEE International Conference on"*, pp. 1916–1921, IEEE, 1996.

[3] S. Balaji, V. Swaminathan, and K. Kannan, "A simple algorithm to optimize maximum independent set," *Advanced Modeling and Optimization*, vol. 12, no. 1, pp. 107–118, 2010.

[4] T. Back and S. Khuri, "An evolutionary heuristic for the maximum independent set problem," in *Proceedings of the "IEEE World Congress on Computational Intelligence"*, pp. 531–535, 1994.

[5] M. Hifi, "A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems," *Journal of the Operational Research Society*, vol. 48, no. 6, pp. 612–622, 1997.

[6] C. C. Aggarwal, J. B. Orlin, and R. P. Tai, "Optimized crossover for the independent set problem," *Operations Research*, vol. 45, no. 2, pp. 226–234, 1997.

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

[8] "The official web-site of DIMACS clique benchmarks." `http://www.cs.hbg.psu.edu/txn131/clique.html`.

Authors

Onur Ugurlu — the 2nd year doctoral student, Faculty of Science, Department of Mathematics, Ege University, Izmir, Turkey; E-mail: *ugurluonur88@gmail.com*

Murat Ersen Berberler — Assistant Professor, Faculty of Science, Department of Computer Science, Dokuz Eylul University, Izmir, Turkey; E-mail: *murat.berberler@deu.edu.tr*

Urfat Nuriyev — Professor, Faculty of Science, Department of Mathematics, Ege University, Izmir, Turkey; E-mail: *urfat.nuriyev@ege.edu.tr*