A NOVEL GENETIC ALGORITHM APPROACH TO THE MAXIMUM INDEPENDENT SET PROBLEM

Mehmet Gencer[1] and Murat Erşen BERBERLER[1]

Abstract. Maximum Independent Set (MIS) is a popular optimization problem of NP-Hard complexity class which is frequently encountered in areas such as image processing, map marking, molecular biology, and scheduling in daily life. In this study, close to optimum quality solutions were searched with genetic algorithm, which is one of the meta-heuristic methods, for MIS problem. Unlike most of the studies in the literature, the initial population of the genetic algorithm was not randomly determined but was formed with the help of various heuristic approaches. Different heuristic approaches and inverse examples in which these approaches do not find optimum value are examined, different solutions have been produced with the help of the shift operator, and the likelihood of heuristic approaches to reach the solution has been increased. By the shift operator is meant the selection of a very small subset of permutation space in order to obtain different solutions which can be generated by the indices vector ordered by a criterion. The algorithm is coded in C and computational experiments are performed on randomly generated graphs with different edge densities. It was observed that the algorithm using the methods proposed in the article is efficient in terms of the values of the run time and objective function.

Keywords: Maximum Independent Set Problem; Genetic Algorithm; Shift Operator

Mathematics Subject Classification. 05C85, 05C69, 90C59, 68R10, 68Q17

...

[1] Faculty of Science, Department of Computer Science, Dokuz Eylul University, 35160, Izmir/TURKEY;
e-mail: gencer.mehmet@ogr.deu.edu.tr & murat.berberler@deu.edu.tr

1. Introduction

In the given $G = (V, E)$ graph, E represents the edges and V represents the vertex. The aim of the problem is to find a V' set which is made $|V'|$ value maximum and conforms with $< i, j > \notin E$ condition (no edge between vertices i and j) for a $V' \subseteq V$ and $\forall i, j \in V'$. In other words, MIS is the problem of finding the largest set of all independent vertices in the graph. The element (vertex) number of this set is called the independence number and is indicated by $\alpha(G)$.The exact finding of the independent set with the maximum element by the enumeration technique requires the examination of all subset selections that match the problem constraints of the given graph. Since the problem is a subset selection problem, it is an optimization problem of the NP-Difficult class.

The MIS (Maximum Independent Set) problem, which is one of the basic problems of graph theory and computational sciences, is encountered in many different areas of human life such as information theory, biology, transportation management, communication, signal processing analysis, classification theory, economics, timelines, experimental design, computer vision and finance [1].

2. Mathematical Model of the Problem

For a given $G = (V, E)$ graph, the number of elements in the set of vertices indicated by V is n, Maximum Independent Set is given with mathematical model;

$$max f(x) = \sum_{i=1}^{n} x_i \tag{1}$$

Such that

$$x_i + x_j \leq 1, \quad \forall (x_i, x_j) \in E \tag{2}$$

and

$$x_i \in \{0, 1\}, \quad i = 1, , n. \tag{3}$$

3. Heuristic Solution Approaches

Heuristic algorithms are often preferred because they are fast and can produce near-optimum solutions in classes with problems of average difficulty. As a subclass of heuristic algorithms, greedy heuristic approaches prefer the most advantageous option at the moment and move on to the next step; In this way, myopic approach can be achieved and in the long term, poor results can be achieved. The most common sequence-based greedy heuristic methods are MAX, MIN and VO (Sequence of vertex degrees). These heuristic methods form the independent set by either accumulating the independent vertex on a set (conservation of the best) or by deleting the connected vertex over the original graph (excretion of the worst). It is applied in some sequence-based heuristic methods for MIS problem. However,

these new approaches often produce reasonable results on some particular graph groups [2].

MAX algorithm is a dynamic algorithm that finds solutions according to the vertex degrees in the graph. In the first step, the vertices and edges that have the highest degree are deleted from the graph. Repeat vertices are counted. The eliminating and recalculation process of degree is continued until there are no vertex connected to each other (edge) in the graph. The remaining independent vertices are given as solutions [3].

MIN algorithm starts to work with an empty I independent set. The algorithm selects the lowest grade vertex in the G graph, adds it to the I set, and deletes it from the G graph. This process continues until there is no vertex in the G graph [4].

Vertex degrees (VO) method, the vertices in the G graph are arranged in a non-decreasing order (in order to be able to describe equality) according to their degree. The first unmarked element of the sequence is taken to the solution, the vertex which is taken to solution and the adjacent vertex of that vertex are marked. This process is continued until all the vertices in the graph are marked [5]. This algorithm is run on the 5 vertex G graph shown in Figure 3.1 and taken as an inverse example. Initially, the solution set (SS) and the set of prohibited vertices (PS) are arranged to empty sets. Vertex degrees of G graph calculated as [1, 2, 2, 2, 3] respectively and $SS = \{1\}$ added to solution because of Vertex with 1 indices in sequence has a value 1 which is the smallest. At this stage, the vertex with 5 indices which is adjacent to peak 1 is marked $PS = \{5\}$ as the forbidden peak. Then $SS = \{1, 2\}$ is included in the solution because of the vertex with the 2 indices in the updated sequence [1*, 2, 2, 2, 3*] has the smallest value 2 in the sequence and vertices with 3 and 4 indices which are adjacent to vertex 2 are marked $PS = \{3, 4, 5\}$ as forbidden peaks. Since calculation is done on all vertices, the algorithm [1 *, 2 *, 2 *, 2 *, 3 *] is terminated and reported as the result of 2 algorithms, the number of elements in the solution set. However, the best solution for this example would be 3, which is the number of element of set includes the vertices 1, 3 and 4.

4. Shift Operator (Carousel Technique)

The MIS problem is a subset selection problem. In order to find the exact solution by enumeration technique, it is necessary to create all possible subsets and to test the suitability of these subsets to the constraints of the problem. 2^n different subsets are created and tested for this operation. Instead of this, the vertices are taken from the sequence of priority formed according to the characteristic of the problem, in order to comply with the problem constraints. Since the sequence is created according to the character of the problem, a solution close to the exact solution is obtained. The situation that precludes the achievement of a full solution is the selection process between vertices with the same priority value. Due to the structure of the problem, the vertices adjacent to the vertices taken before the

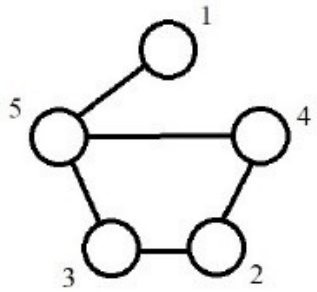

FIGURE 3.1. Example for VO Method

solution are restricted to be included in the solution. In this study, a method has been proposed to overcome such situations and to find values close to the exact solution.

Different permutations are generated by using this sequence in the search for solutions over the priority sequence that is formed according to the characteristic of the problem. Searching solution by sequentially through the sequence corresponds to one of the solutions in the combination space. The property of the method used is search for close solutions of the original sequence, not exceeding its character and the ease of transition between the produced permutations.

Since different permutations are generated from the first sequence of size N, the sequence is written n-1 times consecutively. Starting from the first element, the element is selected from the newly formed n * (n-1) dimension sequence until the n dimension sequence is obtained. This selection is first done as step length 1. This is the amount of progression on the indices of the n * (n-1) dimensional sequence as indicated here. First, starting from the 1st index and since the step length is 1, the elements in the 2nd, 3rd,..,nth indices are taken and correspond to this sequence itself. In the next steps, step amount is increased and the elements in the 1st, 3rd, 5th,..,2n-1.indices are taken. This process is continued until the step length is n-1, in this case the elements in the indices 1st, nth, 2n-1,..,1 + n*(n-1). are taken and the final permutation is produced. Here, n-1 different permutations have been produced and these sequences will be called seed. The new sequences must comply with the permutation rules. Each sequence must be of dimension n and each element must be used once. When coincide with a previously used element, passed to the element in the next index and the step increment continues to be made over the index of this element.

For example, if the first sequence is given as [1, 2, 3, 4, 5], this sequence is written consecutively to obtain [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5] sequence in a dimension of 5 *4. If the 1st index of the initial value is selected, permutations are produced when the step length is 1 [1, 2, 3, 4,5], the step length

is 2 [1, 3, 5, 2, 4], the step length is 3 [1, 4, 2, 5, 3] and finally the step length is 4 [1, 5, 4, 3, 2] .

First Sequence: [1, 2, 3, 4,5]
Production Sequence: [1, 2, 3, 4,5][1, 2, 3, 4,5][1, 2, 3, 4,5][1, 2, 3, 4,5]
(Production Seq.: [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5])
Produced Seed Permutations:
[1, 2, 3, 4, 5]
[1, 3, 5, 2, 4]
[1, 4, 2, 5, 3]
[1, 5, 4, 3, 2]

The reason that these permutations are called seeds is that by using these seeds, number of (n-1) permutations will be produced from each of them. By adding 1 to the index values in the seed, the results to be obtained by starting the process with 2 indices are the same with taking 1 index at first. This is repeated until the addition process (n-1) is made, and in this last step permutation is produced over the element in n. index.

By using first seed [1, 2, 3, 4, 5] produced from [1, 2, 3, 4, 5] sequence, increasing each value by 1 [2, 3, 4, 5, 1] increasing by 2 [3, 4, 5, 1, 2] increasing by 3 [4, 5, 1, 2, 3] and increasing each 4 [5, 1, 2, 3, 4] sequences are produced. This process is repeated on all seeds and as a result 20 permutations will be produced.

Permutations Produced from First Seed
[1, 2, 3, 4, 5]
[2, 3, 4, 5, 1]
[3, 4, 5, 1, 2]
[4, 5, 1, 2, 3]
[5, 1, 2, 3, 4]

With the described process, n*(n-1) permutations are produced. Considering that n! different permutations can be produced in an N dimensional space, it is seen that only permutations belonging to a certain part of space are reached ($\frac{(n(n-1))}{n!}$). As a result of using these permutation sequences, solutions suitable for MIS problem are produced.

In order to reduce the place complexity in the method described, instead of opening n*(n-1) dimensional sequence, the n dimensional sequence is considered to be circular, and it is realized by using the modulation operation according to the n value. Since the process is carried out in the form of rotation on a circle, it is referred to as the carousel technique. Figure 4.1.

5. Genetic Algorithms

Genetic algorithms are search and optimization algorithms that emerge from computer simulation of natural processes. The genetic algorithms that adopt Darwin's principle of survival of individuals who adapt to environmental conditions were first developed by John Holland at the University of Michigan [6]. Holland made these studies to explain the processes of natural systems and to design

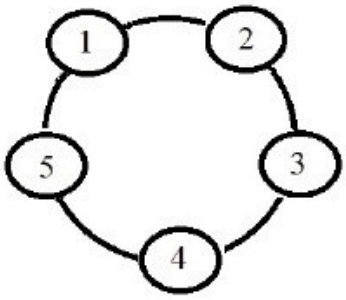

FIGURE 4.1. Shift Operator Production Circle

artificial system software that includes the steps of these systems. This has led to significant innovations in both natural and artificial systems. Genetic algorithms are a search method obtained by applying the principle of conservation of the best and natural selection principle to the computers by simulation. Genetic algorithms were first used to solve the MIS problem by Back and Khuri in 1994 [7].

In this method, work starts with populations of individuals, the individual parameter space represented by the binary sequence represents a point in R^p. In each generation, the value of the objective function for each individual is evaluated as its suitability and a new population is obtained by selecting the more appropriate individuals. Thus, new solutions are created based on the suitability of individuals [8]. Since individuals with high relevance value are often chosen, there is pressure to include more eligible individuals in the population. After several generations, the best individual is expected to represent or at least approach the optimal solution [9].

5.1. Creating the Initial Population

In this study, in contrast to the general approach, prior solutions were not completely randomized when using genetic algorithms. In order to facilitate the solution to achieve better results, the pioneering solutions are produced with heuristic approaches. These approaches are the process of solving the independent vertices by examining the structure of the problem and making a logical decision. [10] The solutions from the heuristic approaches have been improved by using the shift technique and the solutions from this technique have provided diversity in the initial population. In order to increase the diversity in the initial population, some of the solutions were completely random.

5.2. Reproduction Methods Used

The first generation is created in genetic algorithms and ranked according to the conformity function values of these generation chromosomes. The chromosomes with the highest value in the rankings are taken to the new generation without any changes. This process is called elitism. Thanks to elitism, chromosomes with good values are preserved and the new generation will be at least as strong as the previous generation [11]. The chromosomes that are left empty in the new population are formed by crossing from the dual parent chromosomes determined by using roulette wheel technique on all chromosomes of the previous generation.

5.3. Crossing method used

The determination of the individuals to be crossed are realized by the roulette wheel method which is selects according to the suitability value of individuals (chromosomes). The technique used to produce a new chromosome from two parent chromosomes determined by the roulette wheel technique is the process of deciding whether each gene value will come from the mother or father. In this decision-making process, a dice with two values as usual is thrown, if the value of the first chromosome is found according to the dice, the corresponding gene of the first chromosome is taken and in the other case, the corresponding value of the second chromosome is taken. However, a fraudulent dice is used in this process in order to increase the conformity value of the chromosome to be higher. The probability distribution in the dice is directly proportional to the compatibility values of the mother and father chromosomes. In the newly formed chromosome, the relevant gene is more likely to be obtained from the chromosome with high conformity value.

5.4. Mutation

Mutation was applied to prevent the solutions from being attached to local maximums. The implement of mutation to which gene of which chromosome is determined randomly. Since binary coding is used in the solutions, the gene selected in the mutation process is 0 if 1, and 1 if 0. The best results were obtained by using 10% mutation. [12].

5.5. Parameter Values Used

Genetic algorithm, which was created in order to work efficiently on the problem, has many different parameters. These parameters are; population size, maximum number of iterations, crossing rate, mutation rate and stop criteria. [13] While determining these parameters, experiments were conducted on different parameter values as well as literature and tested with problems of different size and density. The size of the population has been kept constant in new generation productions and has led to the destruction of bad value solutions.

6. Calculation Tests

110 problems which has vertex numbers 50, 100, 150, 200, 250, 300, 350, 400, 450 and 500 and has edge density for each vertex number 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 95% are randomly generated. In order to test the success of the genetic algorithm, it is necessary to know the exact solutions of the generated problems. The generated problems were first solved with GAMS-CPLEX and the results are reported in tables in this section. In addition to the GAMS-CPLEX decoder, an intelligent enumeration algorithm is used for dense graphs. Due to the structure of the problems, it was not possible to reach the optimum results of low density problems with edge intensities 5%, 10%, 20%, 30%, in order to test the genetic algorithm, a new set of problems corresponding to said edge intensities has been produced. The optimum values of the new problems were determined during the production of the problems and the ability of the genetic algorithm to reach these values was tested and the results were reported in tables.

6.1. GAMS-CPLEX Calculation Tests

Randomly generated problems were solved with GAMS-CPLEX solver. The optimum values of the problems are given in Table 3 and the working times are given in Table 1 and Table 2. In the tables, n is the size of the problem and d is the edge density value of the problem. The problems indicated by " $>$ " in Table 1 and Table 2 shows the problems that have not been achieved within a reasonable time. Since the optimum value is not found for these problems, they are marked with " $-$ " in Table 3. The said time was determined as 13 hours working time.

Table 1. GAMS Times (sec)

n \ d	5	10	20	30	40	50
50	0,01	0,02	0,05	0,17	0,05	0,03
100	0,12	0,69	1,58	2,81	3,05	2,39
150	2,33	46,72	72,52	50,70	52,05	21,17
200	76,59	1125,37	999,97	442,95	253,11	152,41
250	6132,94	>	>	>	3009,44	1246,92
300	>	>	>	>	>	>
350	>	>	>	>	>	>
400	>	>	>	>	>	>
450	>	>	>	>	>	>
500	>	>	>	>	>	>

TABLE 2. GAMS Times (sec)

n \ d	60	70	80	90	95
50	0,25	0,06	0,03	0,02	0,05
100	2,25	2,42	2,59	1,06	0,39
150	23,22	26,44	26,39	11,47	3,14
200	233,50	318,55	99,530	56,81	18,81
250	773,72	987,39	622,13	150,91	58,44
300	3759,56	1977,97	696,39	183,25	200,28
350	>	3679,44	1771,89	435,72	444,67
400	>	>	4662,31	2177,01	493,23
450	>	>	>	4131,50	942,38
500	>	>	>	9870,61	1974,53

TABLE 3. GAMS Optimum Values

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	42	29	21	14	11	9	7	6	5	4	3
150	51	35	23	17	12	10	9	7	5	4	4
200	60	41	25	18	14	11	9	7	6	4	4
250	68	-	-	-	14	12	9	7	6	4	4
300	-	-	-	-	-	-	10	8	7	5	4
350	-	-	-	-	-	-	-	8	7	5	4
400	-	-	-	-	-	-	-	-	6	5	4
450	-	-	-	-	-	-	-	-	-	5	4
500	-	-	-	-	-	-	-	-	-	5	4

6.2. EXACT SOLUTION ALGORITHM CALCULATION TESTS FOR INTENSE GRAPHS

The GAMS-CPLEX solver may not be efficient in terms of time for some large-scale problems. In order to test the accuracy of the results generated by the genetic algorithm, it is necessary to calculate the optimum values of the problems which are solved. For this reason, an algorithm that makes a complete solution is needed to find the optimum values for the problems. The algorithm, which makes the enumeration process logically and resolves the problems within acceptable times for intense graphs, is run on the problems which are coded and created in C language. The working time of the program is listed in Table 5 and Table 6 and the optimum values for the problems are listed in 4. The problems indicated by " > " in Table 5 and Table 6 shows the problems that have not been achieved because of solution time is more than 10000 sec. Although 2 weeks of working

time is expected for some problems with low edge density in the calculation tests, the algorithm did not finish the case studies and did not reach the result. [14]

Table 4. Algorithm Producing Optimum Solution Results

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	-	-	21	14	11	9	7	6	5	4	3
150	-	-	-	17	12	10	9	7	5	4	4
200	-	-	-	-	14	11	9	7	6	4	4
250	-	-	-	-	14	12	9	7	6	4	4
300	-	-	-	-	15	12	10	8	7	5	4
350	-	-	-	-	16	13	10	8	7	5	4
400	-	-	-	-	16	13	10	8	6	5	4
450	-	-	-	-	16	14	11	9	6	5	4
500	-	-	-	-	17	13	10	9	7	5	4

Table 5. Algorithm Producing Optimum Solution Times

n \ d	5	10	20	30	40	50
50	151,271	3,797	0,093	0,000	0,000	0,000
100	>	>	87,253	1,062	0,078	0,015
150	>	>	>	45,814	1,344	0,124
200	>	>	>	>	11,719	0,610
250	>	>	>	>	54,237	2,375
300	>	>	>	>	284,230	8,797
350	>	>	>	>	1014,426	27,782
400	>	>	>	>	3476,524	68,315
450	>	>	>	>	7392,594	165,366
500	>	>	>	>	>	267,617

All optimum solution values found with GAMS-CPLEX solver and the algorithm that produces complete solution for intense graphs are combined in Table 7. The results of the genetic algorithm will be compared with the values in this table.

6.3. Calculation Tests for Genetic Algorithm

The initial population has been run on randomly generated problems for the work of genetic algorithms created through heuristic solution approaches outside of the general approach in the literature. The genetic algorithm, which contains randomness due to the operators it uses, was executed 100 times for each problem and the resulting minimum, maximum values were given in Table 8 and Table 9 respectively. The average values achieved in 100 times of operation are given in

TABLE 6. Algorithm Producing Optimum Solution Times

n \ d	60	70	80	90	95
50	0,000	0,000	0,000	0,000	0,000
100	0,000	0,000	0,000	0,000	0,000
150	0,015	0,000	0,000	0,000	0,000
200	0,109	0,000	0,000	0,000	0,000
250	0,188	0,031	0,015	0,000	0,000
300	0,641	0,078	0,015	0,000	0,000
350	1,640	0,156	0,015	0,000	0,000
400	3,407	0,297	0,047	0,000	0,000
450	7,109	0,594	0,063	0,015	0,000
500	10,657	0,766	0,079	0,016	0,000

TABLE 7. Optimum Results for Problems

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	42	29	21	14	11	9	7	6	5	4	3
150	51	35	23	17	12	10	9	7	5	4	4
200	60	41	25	18	14	11	9	7	6	4	4
250	68	-	-	-	14	12	9	7	6	4	4
300	-	-	-	-	15	12	10	8	7	5	4
350	-	-	-	-	16	13	10	8	7	5	4
400	-	-	-	-	16	13	10	8	6	5	4
450	-	-	-	-	16	14	11	9	6	5	4
500	-	-	-	-	17	13	10	9	7	5	4

Table 10 and Table 11 and the average operating times in seconds are given in Table 12 and Table 13.

TABLE 8. Genetic Algorithms Minimum Values

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	41	29	20	14	11	9	7	6	5	4	3
150	51	34	22	17	12	10	9	7	5	4	4
200	56	39	24	17	13	11	9	7	6	4	4
250	65	41	25	18	13	11	9	7	6	4	4
300	68	44	26	19	14	12	9	8	7	5	4
350	73	45	26	19	15	12	10	8	7	5	4
400	77	49	28	20	15	12	10	8	6	5	4
450	81	49	28	20	15	12	10	9	6	5	4
500	83	52	29	20	16	13	10	8	7	5	4

TABLE 9. Genetic Algorithms Maximum Values

n \ d	5	10	20	30	40	50	60	70	80	90	95
50	25	20	15	11	10	8	7	5	4	4	3
100	42	29	21	14	11	9	7	6	5	4	3
150	51	35	23	17	12	10	9	7	5	4	4
200	60	41	25	18	14	11	9	7	6	4	4
250	67	44	27	19	14	12	9	7	6	4	4
300	71	46	29	20	15	12	10	8	7	5	4
350	75	48	29	21	16	13	10	8	7	5	4
400	79	53	30	21	16	13	10	8	6	5	4
450	83	53	30	21	16	14	11	9	6	5	4
500	88	54	31	23	17	13	10	9	7	5	4

TABLE 10. Genetic Algorithms Average Values

n \ d	5	10	20	30	40	50
50	25,00	20,00	15,00	11,00	10,00	8,00
100	41,64	29,00	20,75	14,00	11,00	9,00
150	51,00	34,04	22,93	17,00	12,00	10,00
200	59,16	40,59	24,23	17,85	13,84	11,00
250	65,24	41,94	25,94	18,93	13,99	11,93
300	68,80	45,00	27,72	19,16	14,71	12,00
350	74,01	46,73	27,72	19,38	15,59	12,92
400	78,06	51,23	29,28	20,16	15,73	12,70
450	82,44	50,45	29,69	20,62	15,98	13,42
500	86,44	52,95	29,91	21,47	16,10	13,00

TABLE 11. Genetic Algorithms Average Values

n \ d	60	70	80	90	95
50	7,00	5,00	4,00	4,00	3,00
100	7,00	6,00	5,00	4,00	3,00
150	9,00	7,00	5,00	4,00	4,00
200	9,00	7,00	6,00	4,00	4,00
250	9,00	7,00	6,00	4,00	4,00
300	9,98	8,00	7,00	5,00	4,00
350	10,00	8,00	7,00	5,00	4,00
400	10,00	8,00	6,00	5,00	4,00
450	10,94	9,00	6,00	5,00	4,00
500	10,00	8,98	7,00	5,00	4,00

Table 12. Genetic Algorithms Average Times

n \ d	5	10	20	30	40	50
50	0,006	0,007	0,007	0,007	0,005	0,006
100	0,055	0,075	0,084	0,068	0,061	0,050
150	0,174	0,283	0,292	0,231	0,232	0,185
200	0,713	1,019	0,646	0,681	0,541	0,406
250	1,174	1,783	1,785	1,208	1,164	0,887
300	1,501	1,999	2,653	2,382	2,031	1,604
350	3,078	5,610	5,262	3,946	3,052	2,254
400	4,648	7,713	7,102	5,519	4,888	4,580
450	7,271	14,016	10,100	8,773	7,532	6,335
500	8,849	12,771	14,476	11,777	10,34	8,008

Table 13. Genetic Algorithms Average Times

n \ d	60	70	80	90	95
50	0,006	0,006	0,006	0,004	0,004
100	0,058	0,056	0,050	0,043	0,047
150	0,152	0,144	0,175	0,161	0,103
200	0,375	0,340	0,371	0,364	0,243
250	0,819	0,781	0,737	0,748	0,513
300	1,340	1,300	0,969	1,170	0,951
350	2,573	2,074	1,554	1,708	1,572
400	3,433	2,959	3,460	2,829	2,620
450	4,853	4,038	5,050	4,035	4,037
500	10,558	5,811	5,099	6,011	5,692

In order to see the success of the genetic algorithms created by heuristic method, the average values of the solved problems (Table 10 and Table 11) and the optimum values for these problems (Table 7) should be examined. The average values mentioned are the averages of the values found by running the program 100 times. The relative error tables in Table 14 and Table 15 were created using the average values reached by the genetic algorithm in problems and the optimum values for problems.

6.4. Additional Problems with Low Density

Since the randomly generated low density problems (5%, 10%, 20%, 30%) could not be fully solved by the efficient enumeration algorithm and GAMS, the comparative performance ratio of the proposed genetic algorithm could not be calculated. In order to eliminate this handicap, low-density problems were not generated randomly but were obtained according to a certain systematic: Firstly, the number of edges of the related density was obtained depending on the number of vertices.

TABLE 14. Genetic Algorithms Relative Error

n \ d	5	10	20	30	40	50
50	0,000	0,000	0,000	0,000	0,000	0,000
100	0,009	0,000	0,012	0,000	0,000	0,000
150	0,000	0,027	0,003	0,000	0,000	0,000
200	0,014	0,010	0,031	0,008	0,011	0,000
250	0,041	-	-	-	0,001	0,006
300	-	-	-	-	0,019	0,000
350	-	-	-	-	0,026	0,006
400	-	-	-	-	0,017	0,023
450	-	-	-	-	0,001	0,041
500	-	-	-	-	0,053	0,000

TABLE 15. Genetic Algorithms Relative Error

n \ d	60	70	80	90	95
50	0,000	0,000	0,000	0,000	0,000
100	0,000	0,000	0,000	0,000	0,000
150	0,000	0,000	0,000	0,000	0,000
200	0,000	0,000	0,000	0,000	0,000
250	0,000	0,000	0,000	0,000	0,000
300	0,002	0,000	0,000	0,000	0,000
350	0,000	0,000	0,000	0,000	0,000
400	0,000	0,000	0,000	0,000	0,000
450	0,005	0,000	0,000	0,000	0,000
500	0,000	0,002	0,000	0,000	0,000

Then, in order to obtain this number of edge in total, the vertex number of the exact graphs which are known to be the maximum independent number and which are the highest number of edges are isolated from each other is calculated. The matrix of the problem is filled with isolated whole graphs (for example k pcs) based on the calculated number of vertices. After this process, if there is an idle vertex (for example m pcs), the sample problem is obtained with a road graph containing the remaining vertices. The exact solution of the sample problem is that the maximum independence of the complete line and the path line is 1 and $\lceil m/2 \rceil$, respectively, so that $k + \lceil m/2 \rceil$. In addition, the best solution of the relevant sample problem is written on the last line of the text file of the problem. Genetic algorithm was run 100 times on the generated problems and average run times in Table 16 were obtained. Optimum values have been solved 100 times with genetic algorithms with problems given in Table 17 and it has been seen that genetic algorithms can reach these values. The heuristic approach in the algorithm showed good success on these problems and the optimum values could be calculated after the genetic algorithm processes.

Table 16. Low Density Additional Problems GA Average Times

n \ d	5	10	20	30
50	0,004	0,003	0,004	0,003
100	0,026	0,023	0,021	0,025
150	0,080	0,081	0,085	0,095
200	0,191	0,183	0,172	0,232
250	0,356	0,442	0,326	0,295
300	0,713	0,619	0,735	0,554
350	1,704	1,680	1,496	1,256
400	2,212	2,329	2,750	1,704
450	3,364	3,156	2,789	2,094
500	4,594	3,625	5,857	2,734

Table 17. Low Density Additional Problems GA Optimum Values

n \ d	5	10	20	30
50	15	9	7	6
100	18	10	8	11
150	23	12	10	15
200	19	15	13	19
250	21	17	13	5
300	24	20	15	6
350	23	15	18	5
400	20	15	20	6
450	26	15	23	7
500	22	15	25	8

7. Conclusions

In this study, GAMS-CPLEX solver, which is one of the low runtime ready-made software and which guarantees the optimum solution based on the mathematical model of the problem, is used in the solution of the maximum independent set problem from NP-Hard class problems. Since this solver solves the problem with the branch-bound method, it is possible to reach the optimum result by experimenting faster on the possible solutions. The GAMS-CPLEX solver achieved optimum results in a very short time without any problems in randomly generated problems in the first part of the study. However, we faced different problems while solving the real problems of the thesis. The first of these problems is that the length of the solution is taking too long for problems with an edge density of 40 % or less, which is more likely to be tried. The second problem was that CPLEX's solution was based on the RAM of the computer and on the hard disk when RAM was not enough. As the number of cases to be tried increases, the

solution tree used by the program reaches a very large size, restricting the operation of the computer and giving the GAMS-CPLEX solver memory error before the problem is solved. For these reasons, in order to reach optimum solutions, full solution program is used for dense graphs written in C language. Due to the structure of the MIS problem, this program easily found the optimum values for the problems of dense graphs, but had problems in reaching the optimum values even for the second smallest problem size n = 100 in the sparse graphs. In the next step, genetic algorithms were used with meta-heuristic approach, which gave close to optimum results in short periods of time. In order to test genetic algorithms that do not guarantee optimum discovery, problems with known optimal values are needed. The optimum value could not be calculated for the low density of the 110 problems. For this reason, in order to test genetic algorithms on sparse problems, a group of problems whose predetermined values have been determined has been formed. For sparse graphs, a complete solution algorithm is proposed in the following studies. In the main focus of the study, the initial population of the genetic algorithm was created by using the heuristic solution approaches in literature and the shifting technique which can produce different results using the results obtained from these approaches. In order to make the comparison clearly, the relative errors of the results obtained by the genetic algorithm in the problems were calculated using the optimum results of the problems. 110 first group, 40 additional problem sets, 150 problems and working time of algorithms are reported with tables. Genetic algorithm was run 100 times for each problem and minimum, maximum and average tables of values were created accordingly. Genetic algorithm with heuristic approach was found to be successful in finding optimum results and approaching optimum values.

References

[1] Butenko, S. "Maximum independent set and related problems, with applications," PhD thesis, University of Florida, 2003.

[2] N. C. Lê, C. Brause and I. Schiermeyer, "On sequential heuristic methods for the maximum independent set problem," *Discussiones Mathematicae Graph Theory,37*(2), 415-426, 2017.

[3] J. R. Griggs, "Lower bounds on the independence number in terms of the degrees," *Journal of Combinatorial Theory, Series B, 34*(1), 22-39, 1983.

[4] O. J. Murphy, "Lower Bounds on the Stability Number of Graphs Computed in Terms of Degrees," *Discrete Mathematics 90*(2), 207-211, 1991.

[5] N. V. Mahadev, B. A. Reed, "A note on vertex orders for stability number," *Journal of Graph Theory, 30*(2), 113-120, 1999.

[6] J. H. Holland, "Adaptation in natural and artificial systems," *MI: The University of Michigan Press*, Ann Arbor.

[7] T. Back, and S. Khuri, An evolutionary heuristic for the maximum independent set An evolutionary heuristic for the maximum independent set problem. *Proceedings of the First IEEE Conference on Evolutionary Computation*, Vol. 2. Orlando, Florida, USA: IEEE, Jan. 1994, pp. 531-535. ISBN: 0-7803-1899-4.

[8] T. Cura, "Modern sezgisel teknikler ve uygulamalar," *Papatya Yaynclk Eitim*, 2008.

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

[10] M. Hifi, "A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems." *Journal of the Operational Research Society, 48*(6), 612-622, 1997.

[11] C. W. Ahn, R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Transactions on Evolutionary Computation, 7*(4), 367-385, 2003.

[12] I. De Falco, A.D. Cioppab, E. Tarantinoa, "Mutation-based genetic algorithm: performance evaluation", *Applied Soft Computing*, Volume 1, Issue 4, p.p. 285-299, May 2002.

[13] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on systems, man, and cybernetics, 16*(1), 122-128, 1986.

[14] E. Nasibov, M. Berberler, C. Atilgan, "An efficient algorithm for exact solution of maximum independent set problem in dense graphs," *International Symposium on Computing in Science Engineering. Proceedings*, 2013.