Advances in Dynamical Systems and Applications (ADSA).
ISSN 0973-5321, Volume 17, Number 1, (2022) pp. 331-343

http://www.ripublication.com/adsa.htm

A Modified Discrete Differential Evolution Algorithm for Solving Graph Coloring Problem

Kanita Panhong[1], **Tawun Remsungnen**[2],
Jeerayut Wetweerapong[1], **Pikul Puphasuk**[1,*]

[1]*Department of Mathematics, Faculty of Science*
Khon Kaen University, Khon Kaen, 40002, Thailand.
[2]*Faculty of Interdisciplinary Studies, Khon Kaen University*
Nong Khai Campus, Nong Khai, 43000, Thailand.

Abstract

Graph coloring problem (GCP) is a combinatorial optimization problem with many applications in science and engineering. Because the coloring is sensitive to the order of vertices and is computationally hard, heuristic search methods on the domain of permutation of vertices have become a practical solution approach. In this work, we propose a modified discrete differential algorithm (MDDE) to generate suitable permutations by improving mutation and crossover operators. The method is tested and compared with several well-known methods using DIMACS benchmark graphs. The experimental results show that the proposed MDDE is effective for the graph coloring problem.

Keywords: Discrete differential evolution algorithm, permutation-based combinatorial optimization problems, graph coloring problem, optimization.

1. INTRODUCTION

Permutation-based combinatorial optimization problems arise in many fields of science and engineering. They include vehicle routing problem, flow shop scheduling problem, optical permutation network, traveling salesman problem, and graph coloring problem. Since they are NP-hard problems, many researchers have studied and developed heuristic algorithms to search for optimum solutions. Some well-known algorithms are genetics algorithm

(GA) [1], branch and bound algorithm (BB) [2], tabu search algorithm (TS) [3], particle swarm optimization algorithm (PSO) [4] and differential evolution algorithms (DE) [5].

The DE algorithm is an efficient evolutionary algorithm for solving continuous optimization problems. Its discrete version has been designed and proposed for solving combinatorial optimization problems. This research focuses on solving the graph coloring problem (GCP), which has many applications such as timetable scheduling [6], examination scheduling [7], and channel routing [8]. Let $G = (V, E)$ be a graph with a vertex set V and an edge set E. The k coloring is a function $c : V \to \{1, 2, \ldots, k\}$ such that $c(u) \neq c(v)$ for each $(u, v) \in E$. A graph G is k colorable if it has a k coloring. The graph coloring problem finds the minimum number k that G is k colorable. This number is called the chromatic number of G denoted by $\chi(G)$.

There are many evolutionary algorithms proposed for solving the GCP. In 2008, Bui et al. [9] proposed an ant-based algorithm (ABAC). Unlike the ant colony optimization (ACO), each ant in ABAC colors only a portion of the graph. The results show that ABAC performed well on some benchmark graphs. In 2010, Ray et al. [10] proposed the MSPGCA algorithm with a new operator called the double point guided mutation to improve the performance of the simple genetic algorithm GAGCA. Then, Faraji and Javadi [11] proposed an algorithm based on bees' behavior in nature called BEECOL in 2011. The results show that BEECOL outperforms the Max-Min ant system algorithm (MMGC). In 2012, Pal et al. [12] presented the modification of the simulated annealing method called MSAGCP, which performs better than the simulated annealing method SAGCP. The MSAGCP can find the best-known results for most of the tested graphs but still fails for some instances. In 2015, Mahmoudi and Lotfi [13] introduced a modified cuckoo optimization algorithm called MCOACOL. It is tested on several graph coloring benchmark problems and compared with some well-known heuristic search methods. The obtained results show the high performance of MCOACOL.

In this research, we propose a modified discrete differential algorithm (MDDE) that improves mutation and crossover operations for solving the graph coloring problem. The remainder of the paper is organized as follows. Section 2 presents the algorithm descriptions of the Welsh-Powell method, differential evolution algorithm, and the proposed MDDE method. Section 3 explains the experimental designs. Section 4 presents the performance comparisons of MDDE and some well-known methods. Then the conclusion is given in the last section.

2. ALGORITHM DESCRIPTIONS

2.1. Welsh-Powell method

Since the GCP is NP-hard, the use of heuristic techniques is necessary. One of the effective heuristic methods to color the vertices of graph G is the Welsh-Powell method [14]. Let $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set and $E = V \times V$ is the edge set. The Welsh-Powell algorithm can be described as follows:

Step 1: List the vertices in order of descending degrees.

Step 2: Color the first vertex in the list (the vertex with the maximum degree) with color 1.

Step 3: Move down the list and color all the vertices not connected to the colored vertex with the same color.

Step 4: Repeat Step 3 on all uncolored vertices with a new color in descending order of degrees until all the vertices are colored.

2.2. Differential evolution algorithm

2.2.1 DE for continuous optimization problems

The DE algorithm is an efficient population-based optimization method for continuous optimization problems proposed by Storn and Price in 1997 [5]. It consists of three basic population operations: mutation, crossover, and selection. First, the initial population of real vectors is generated uniformly in the feasible region. For each generation and each target vector x_i, a mutant vector $v_i = x_{r_1} + F(x_{r_2} - x_{r_3})$ is constructed from three different random population vectors x_{r_1}, x_{r_2} and x_{r_3} which are also different from x_i where F is a scaling factor. The components of v_i are exchanged with those of the target vector x_i according to the crossover rate CR to construct a trial vector u_i. Then, the target vector x_i is replaced by the trial vector u_i if u_i is better than x_i. DE population vectors will evolve iteratively and move toward an optimal solution.

2.2.2 DE for permutation-based optimization problems

The DE algorithm for permutation-based optimization problems needs some modifications. Since all components of a population vector are distinct integers, we cannot use the difference of vectors directly. However, the property that two vectors have some equal component values is essential for the mutation operation. The crossover operation also needs to be modified in designing an efficient permutation-based DE algorithm.

2.3. The proposed MDDE method

Since the heuristic methods to color the vertices are sensitive to the order of vertices, we propose a modified discrete differential algorithm (MDDE) to generate suitable permutations by improving mutation and crossover operators to balance exploration and exploitation during the search. The mutation strategy considers the distances between corresponding components of two permutation vectors and uses random insertion operation to generate a mutant vector. To accelerate the search, MDDE uses a small proportion of mutant permutations as trial permutations. For the usual mutant permutations, the crossover strategy applies the two-cut operator [15] to exchange their contents with the target permutation to construct the trial permutation. The proposed MDDE is described as follows.

Step 1 *Input*: A graph $G = (V, E)$, objective function to be minimized (f), problem dimension (D), population size (NP), maximum number of generations (MG), mutation parameter F in $[0, 1]$, and crossover probabilities Pc and Pt in $[0, 1]$.

Step 2 *Initialization*: Randomly generate the initial population of NP permutations on D vertices. Color the vertices of G by using each permutation and the Welsh-Powell method. Find the best permutation $xbest$ and its best value $fbest$ (number of colors).

Step 3 *Mutation*: For each target permutation x_i, construct the mutant permutation v_i by the following steps.

3.1 Calculate the auxiliary vector δ_i by

$$\delta_{i,j} = \begin{cases} x_{r_1,j} \,;\, x_{r_2,j} - x_{r_3,j} = 0 \text{ or } \mid \dfrac{x_{r_2,j} - x_{r_3,j}}{D} \mid \geq F \\ 0 \quad ;\, otherwise \end{cases} \tag{2.1}$$

where $j = 1, 2, \ldots, D$ and $x_{r_1}, x_{r_2}, x_{r_3}$ are different random population permutations which are also different from the target x_i. They are sorted as

$$f(x_{r_1}) \leq f(x_{r_2}) \leq f(x_{r_3}).$$

3.2 Random a permutation vector y. Remove the components of y such that $\delta_{i,j}$ are not equal to 0 and let $\bar{y}$ be the vector of the remaining components of y. Then insert each component of $\bar{y}$ into each component of δ_i such that $\delta_{i,j} = 0$ to obtain a mutant permutation v_i.

Step 4 *Crossover*: Construct the trial permutation u_i using consecutive parts of x_i and v_i as follows.

4.1 Random a number r in $(0, 1)$.
4.2 If $r \geq Pc$, then let $u_i = v_i$.
4.3 If $r < Pc$, then exchange the contents of x_i and v_i to obtain u_i. Random a position $k \in \{1, 2, \ldots, D\}$ to cut x_i into 2 parts. Set $p_1 = [x_{i,1}, x_{i,2}, ..., x_{i,k}]$. Remove components of p_1 from v_i and let p_2 be a vector of the remaining components. Construct z_1 and z_2 by concatenating p_1 and p_2 as $z_1 = p_1 p_2$ and $z_2 = p_2 p_1$. The trial permutation u_i is obtained by

$$u_i = \begin{cases} z_1\,;\ s < Pt \\ z_2\,;\ otherwise \end{cases} \tag{2.2}$$

where s is a random number in $[0, 1]$.

Step 5 *Selection*: Apply the greedy selection to select the population permutation for the next generation by

$$x_i = \begin{cases} u_i & ;\ f(u_i) \leq f(x_i) \\ x_i & ;\ otherwise \end{cases} \tag{2.3}$$

Update the best permutation $xbest$ and the current minimum number of colors $fbest$.

Step 6 *Stopping condition*: Repeat all Steps 3 - 5 until reaching the MG. Report the obtained best permutation $xbest$ and the best value $fbest$.

The flowchart of the MDDE method is illustrated in Figure 1.

3. EXPERIMENTAL DESIGNS

We conduct three experiments to assess the performance of the MDDE against several well-known methods. The first experiment compares MDDE with GAGCA and MSPGCA algorithms [10]. The second experiment compares MDDE with SAGCP and MSAGCP algorithms [12]. Then the third experiment compares MDDE with MCOACOL, ABAC, and BEECOL algorithms [13]. The parameters $F = 0.5, Pc = 0.8, Pt = 0.8, NP = 50$, and $MG = 100$ are set for MDDE, and the algorithm is performed 50 independent runs for each graph. The settings of the compared methods are taken from the original papers. All methods are tested on some selected graphs of the DIMACS benchmark downloaded from the home page: https://mat.gsia.cmu.edu/COLOR/instances.html. The example graphs are illustrated in Figure 2 by using Social Network Visualizer Software [16].

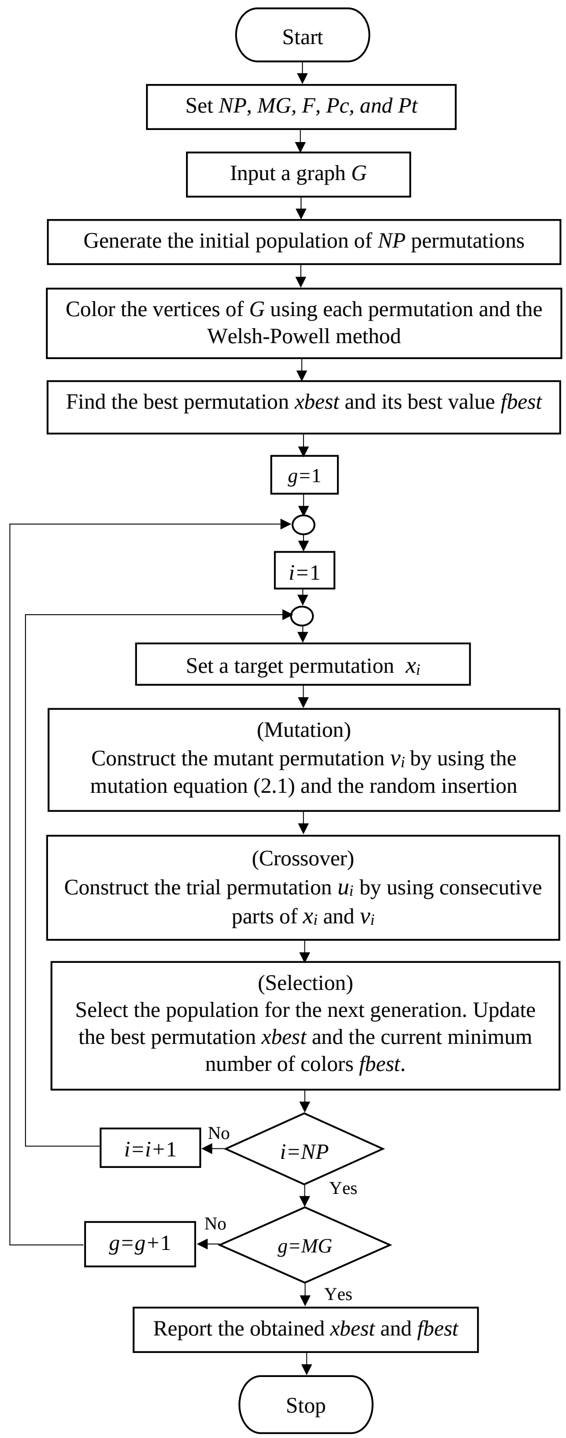

Figure 1. Flowchart of the proposed MDDE method.

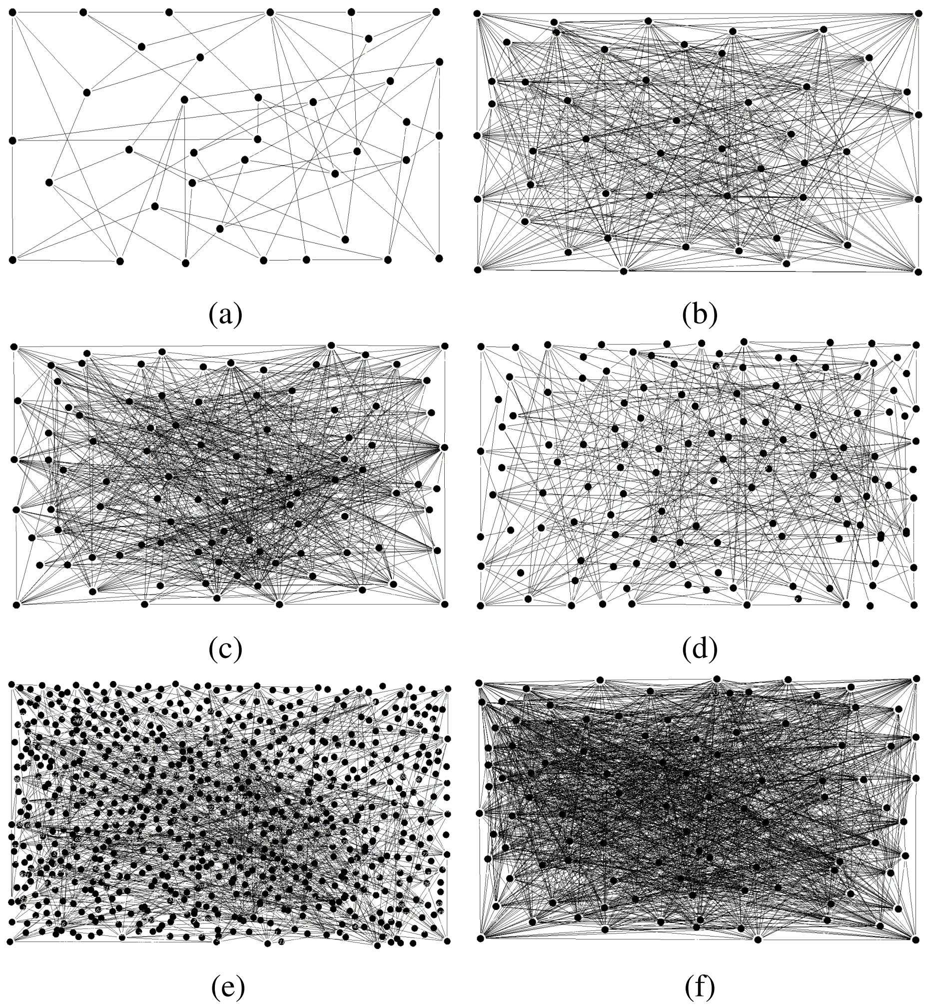

Figure 2. The example graphs: (a) 2-Insertion_3, (b) queen7_7, (c) myciel6, (d) miles250, (e) DSJCI 125.1, and (f) queen10_10

4. RESULTS AND DISCUSSION

This section presents the performance comparisons of MDDE with the compared methods in Table 1 - Table 3. The first four columns of each table show the name of graphs, number of vertices, number of edges, and the chromatic numbers of graphs. The last columns show the best values obtained by MDDE and the compared methods. The tables list the test graphs in the order of the numbers of vertices. We indicate the minimum numbers of colors obtained among the algorithms in bold. The last row summarizes the total cases that each algorithm give the numbers of colors equal to $\chi(G)$.

Table 1. Performance comparison of the MDDE, MSPGCA and GAGCA.

Graphs	Vertices	Edges	$\chi(G)$	MDDE	MSPGCA [10]	GAGCA [10]
queen5_5	25	320	5	**5**	**5**	6
queen6_6	36	580	7	**7**	8	9
myciel5	47	236	6	**6**	**6**	7
queen7_7	49	952	7	**7**	**7**	12
queen8_8	64	1456	9	**9**	11	15
huck	74	301	11	**11**	**11**	**11**
jean	80	508	10	**10**	**10**	11
queen9_9	81	2112	10	**10**	**10**	19
david	87	406	11	**11**	**11**	**11**
mugg88_25	88	146	4	**4**	**4**	**4**
myciel6	95	755	7	**7**	**7**	10
queen8_12	96	2736	12	**12**	14	23
mugg100_25	100	166	4	**4**	**4**	5
queen10_10	100	2940	11	**12**	14	20
4-FullIns_3	114	541	7	**7**	**7**	9
Games120	120	638	9	**9**	**9**	**9**
DSJCI 125.1	125	736	5	**6**	**6**	8
miles750	128	2113	31	**31**	**31**	34
miles1000	128	3216	42	**42**	**42**	45
miles1500	128	5198	73	**73**	**73**	**73**
anna	138	986	11	**11**	**11**	**11**
2-Insertions_4	149	541	5	**5**	**5**	**5**
5-FullIns_3	154	792	8	**8**	**8**	9
myciel7	191	2360	8	**8**	**8**	32
mulsol.i.1	197	3925	49	**49**	**49**	52
1-Insertions_5	202	1227	6	**6**	**6**	7
2-FullIns_4	212	1621	6	**6**	**6**	8
3-Insertions_4	281	1046	5	**5**	**5**	7
1-FullIns_5	282	3247	6	**6**	**6**	7
3-FullIns_4	405	3524	7	**7**	**7**	8
4-Insertions_4	475	1795	5	**5**	**5**	8
homer	561	1629	13	**13**	**13**	15
Total cases with the numbers of colors equal to $\chi(G)$				31 cases	28 cases	7 cases

4.1. Performance comparison of MDDE, MSPGCA and GAGCA

The performances of MDDE on 32 test graphs are compared with those of MSPGCA and GAGCA as reported in [10]. Table 1 shows that MDDE gives the numbers of colors equal to $\chi(G)$ for 30 graphs with the remaining numbers different from $\chi(G)$ only 1. The reported numbers of colors by GAGCA and MSPGCA equal to $\chi(G)$ for 7 cases and 28 cases, respectively. Their remaining numbers differ from $\chi(G)$ between 1 to 11. It indicates that MDDE significantly outperforms GAGCA and slightly outperforms MSPGCA.

Table 2. Performance comparison of MDDE, MSAGCP and SAGCP.

Graphs	Vertices	Edges	$\chi(G)$	MDDE	MSAGCP [12]	SAGCP [12]
queen5_5	25	320	5	**5**	**5**	7
queen7_7	49	952	7	**7**	**7**	9
huck	74	301	11	**11**	**11**	**11**
jean	80	508	10	**10**	**10**	11
queen9_9	81	2112	10	**10**	11	14
david	87	406	11	**11**	**11**	**11**
mugg88_25	88	146	4	**4**	**4**	**4**
myciel6	95	755	7	**7**	**7**	8
mugg100_25	100	166	4	**4**	**4**	**4**
queen10_10	100	2940	11	**12**	**12**	17
4-FullIns_3	114	541	7	**7**	**7**	**7**
Games120	120	638	9	**9**	**9**	10
DSJCI 125.1	125	736	5	**6**	**6**	10
miles750	128	2113	31	**31**	**31**	35
miles1000	128	3216	42	**42**	45	50
Miles1500	128	5198	73	**73**	75	80
anna	138	986	11	**11**	**11**	13
2-Insertions_4	149	541	5	**5**	**5**	6
5-FullIns_3	154	792	8	**8**	**8**	**8**
mulsol.i.1	197	3925	49	**49**	52	58
1-Insertions_5	202	1227	6	**6**	7	9
Zeroin_i_2	211	3541	30	**30**	**30**	44
2-FullIns_4	212	1621	6	**6**	**6**	**6**
3-Insertions_4	281	1046	5	**5**	**5**	**5**
1-FullIns_5	282	3247	6	**6**	**6**	8
3-FullIns_4	405	3524	7	**7**	9	12
4-Insertions_4	475	1795	5	**5**	6	7
homer	561	1629	13	**13**	**13**	17
Total cases with the numbers of colors equal to $\chi(G)$				26 cases	19 cases	8 cases

4.2. Performance comparison of MDDE, MSAGCP and SAGCP

Table 2 presents the performance comparisons of MDDE, MSAGCP, and SAGCP on 28 graphs. They obtain the numbers of colors equal to $\chi(G)$ for 26, 19, and 8 cases, respectively. The MDDE gives smaller numbers of colors than those of MSAGCP and SAGCP for 18 and 6 graphs, respectively. This shows that the proposed MDDE outperforms both SAGCP and MSAGCP.

4.3. Performance comparison of MDDE, MCOACOL, ABAC and BEECOL

Table 3 presents the performance comparison of MDDE, MCOACOL, ABAC, and BEECOL on 63 graphs. The results show that they give the numbers of colors equal to $\chi(G)$ for 61, 57, 62, and 57 graphs, respectively. Their remaining numbers differ from $\chi(G)$ only 1. Thus, all methods are effective for the GCP. In addition, the proposed MDDE and ABAC slightly outperform MCOACOL and BEECOL.

Table 3. Performance comparison of MDDE, MCOACOL , ABAC, and BEECOL.

Graphs	Vertices	Edges	$\chi(G)$	MDDE	MCOACOL [13]	ABAC [13]	BEECOL [13]
Myciel3	11	20	4	**4**	**4**	**4**	**4**
myciel4	23	71	5	**5**	**5**	**5**	**5**
queen5_5	25	320	5	**5**	**5**	**5**	**5**
1-FullIns_3	30	100	4	**4**	**4**	**4**	**4**
queen6_6	36	580	7	**7**	8	**7**	8
2-Insertions_3	37	72	4	**4**	**4**	**4**	**4**
myciel5	47	236	6	**6**	**6**	**6**	**6**
queen7_7	49	952	7	**7**	**7**	**7**	8
2-FullIns_3	52	201	5	**5**	**5**	**5**	**5**
3-Insertions_3	56	110	4	**4**	**4**	**4**	**4**
queen8_8	64	1456	9	**9**	10	**9**	10
1-Insertions_4	67	232	5	**5**	**5**	**5**	**5**
huck	74	301	11	**11**	**11**	**11**	**11**
4-Insertions_3	79	156	4	**4**	**4**	**4**	**4**
jean	80	508	10	**10**	**10**	**10**	**10**
3-FullIns_3	80	346	6	**6**	**6**	**6**	**6**
queen9_9	81	2112	10	**10**	11	**10**	11
david	87	406	11	**11**	**11**	**11**	**11**
mugg88_1	88	146	4	**4**	**4**	**4**	**4**
mugg88_25	88	146	4	**4**	**4**	**4**	**4**
1-FullIns_4	93	593	5	**5**	**5**	**5**	**5**
myciel6	95	755	7	**7**	**7**	**7**	**7**
queen8_12	96	2736	12	**12**	13	**12**	**12**

Table 3. Performance comparison of MDDE, MCOACOL , ABAC, and BEECOL (Cont.).

Graphs	Vertices	Edges	$\chi(G)$	MDDE	MCOACOL [13]	ABAC [13]	BEECOL [13]
mugg100_1	100	166	4	**4**	**4**	**4**	**4**
mugg100_25	100	166	4	**4**	**4**	**4**	**4**
queen10_10	100	2940	11	12	12	**11**	12
4-FullIns_3	114	541	7	**7**	**7**	**7**	**7**
Games120	120	638	9	**9**	**9**	**9**	**9**
DSJCI 125.1	125	736	5	**6**	**6**	**6**	**6**
miles250	128	387	8	**8**	**8**	**8**	**8**
miles500	128	1170	20	**20**	**20**	**20**	**20**
miles750	128	2113	31	**31**	**31**	**31**	**31**
miles1000	128	3216	42	**42**	**42**	**42**	**42**
Miles1500	128	5198	73	**73**	**73**	**73**	**73**
anna	138	986	11	**11**	**11**	**11**	**11**
2-Insertions_4	149	541	5	**5**	**5**	**5**	**5**
5-FullIns_3	154	792	8	**8**	**8**	**8**	**8**
mulsol.i.2	188	3885	31	**31**	**31**	**31**	**31**
myciel7	191	2360	8	**8**	**8**	**8**	**8**
mulsol.i.1	197	3925	49	**49**	**49**	**49**	**49**
1-Insertions_5	202	1227	6	**6**	**6**	**6**	**6**
Zeroin_i_3	206	3540	30	**30**	**30**	**30**	**30**
Zeroin_i_1	211	4100	49	**49**	**49**	**49**	**49**
Zeroin_i_2	211	3541	30	**30**	**30**	**30**	**30**
2-FullIns_4	212	1621	6	**6**	**6**	**6**	**6**
3-Insertions_4	281	1046	5	**5**	**5**	**5**	**5**
1-FullIns_5	282	3247	6	**6**	**6**	**6**	**6**
3-FullIns_4	405	3524	7	**7**	**7**	**7**	**7**
Fpsol2.i.3	425	8688	30	**30**	**30**	**30**	**30**
le450 25a	450	5714	25	**25**	**25**	**25**	**25**
le450 25b	450	8263	25	**25**	**25**	**25**	**25**
Fpsol2.i.2	451	8691	30	**30**	**30**	**30**	**30**
4-Insertions_4	475	1795	5	**5**	**5**	**5**	**5**
Fpsol2.i.1	496	11654	65	**65**	**65**	**65**	**65**
homer	561	1629	13	**13**	**13**	**13**	**13**
2-Insertions_5	597	3936	6	**6**	**6**	**6**	**6**
1-Insertions_6	607	6337	7	**7**	**7**	**7**	**7**
Inithx.i.3	621	13969	31	**31**	**31**	**31**	**31**
Inithx.i.2	645	13979	31	**31**	**31**	**31**	**31**
4-FullIns_4	690	6650	8	**8**	**8**	**8**	**8**
2-FullIns_5	852	12201	7	**7**	**7**	**7**	**7**
Inithx.i.1	864	18707	54	**54**	**54**	**54**	**54**
5-FullIns_4	1085	11395	9	**9**	**9**	**9**	**9**
Total cases with the numbers of colors equal to $\chi(G)$				61 cases	57 cases	62 cases	57 cases

5. CONCLUSION

In this research, we proposed a discrete differential evolution algorithm MDDE to solve the graph coloring problems. The MDDE uses new permutation-based mutation and crossover operators to balance diversification and intensification during the search. Extensive experiments show that the proposed MDDE is effective for solving the GCP. Future research can investigate the possibility of solving other graph problems such as maximum clique problems and the maximum independent set problems using the same approach.

ACKNOWLEDGEMENT

This research is supported by Department of Mathematics, Faculty of Science, Khon Kaen University, Fiscal Year 2022.

REFERENCES

[1] Houck, C. R., Joines, J., and Kay, M. G., 1995, "A genetic algorithm for function optimization: A Matlab implementation", Ncsu-ie tr, 95, pp. 1-10.

[2] Ariano, A. D', Pacciarelli, D., and Pranzo, M., 2007, "A branch and bound algorithm for scheduling trains in a railway network", European Journal of Operational Research, 183, pp. 643-657.

[3] Nowicki, E., and Smutnicki, C., 1996, "A fast tabu search algorithm for the permutation flow-shop problem", European Journal of Operational Research, 91, pp. 160-175.

[4] Kennedy, J., and Eberhart, R., 1995, "Particle swarm optimization", Proc. IEEE. Int. Conf. Neural Network, 4, pp. 1942-1948.

[5] Storn, R., and Price, K., 1997, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Pptimization, 11, pp. 341-359.

[6] Wood, D. C., 1969, "A technique for coloring a graph applicable to large scale time-tabling", Computer Journal, 12, pp. 317-319.

[7] Leighton, F. T., 1979, "A graph coloring algorithm for large scheduling problems", Journal of Research of the National Bureau of Standards, 84(6), pp. 489-505.

[8] Sarma, S. S., Mondal, R., and Seth, A., 1985, "Some sequential graph coloring algorithms for restricted channel routing", INT. J. Electronics, 77(1), pp. 81-93.

[9] Bui, T. N., Nguyen, T. H., Patel, C. M., and Phan, K. A. T., 2008, "An ant-based algorithm for coloring graphs", Discrete Applied Mathematics, 156(2), pp. 190-200.

[10] Ray, B., Pal, A. J., Bhattacharyya, D., and Kim, T., 2010, " An efficient GA with multipoint guided mutation for graph coloring problems", International Journal of Signal Processing, Image Processing and Pattern Recognition, 3(2), pp. 51-58.

[11] Faraji, M., and Haj Seyyed Javadi, H., 2011, "Proposing a new algorithm based on bees behavior for solving graph coloring", International Journal of Contemporary Mathematical Sciences, 6(1), pp. 41-49.

[12] Pal, A. J., Ray, B., Zakaria, N., and Sarma, S. S., 2012, "Comparative performance of modified simulated annealing with simple simulated annealing for graph coloring problem", Procedia Computer Science, 9, pp. 321-327.

[13] Mahmoudi, S., and Lotfi, S., 2015, "Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem", Applied Soft Computing, 33, pp. 48-64.

[14] Welsh, D. J. A., and Powell, M. B., 1967, "An upper bound for the chromatic number of a graph and its application to timetabling problems", The Computer Journal, 10(1), pp. 85-86.

[15] Pan, Q. K., Tasgetiren, M. F., and Liang, Y. C., 2008, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem", Computers & Industrial Engineering, 55(4), pp. 795-816.

[16] SocNetV: Social Network Analysis and Visualization Software: https://socnetv.org.