Memetic Algorithm with Filtering Scheme for the Minimum Weighted Edge Dominating Set Problem

Shada N. Abdel-Aziz
Dept. of Computer Science
Faculty of Computers & Information
Assiut, Egypt

Abdel-Rahman Hedar
Dept. of Computer Science
Faculty of Computers & Information
Assiut, Egypt
Email: hedar@aun.edu.eg

Adel A. Sewisy
Dept. of Computer Science
Faculty of Computers & Information
Assiut, Egypt

***Abstract*—The minimum weighted edge dominating set problem (MWEDS) generalizes both the weighted vertex cover problem and the problem of covering the edges of graph by a minimum cost set of both vertices and edges. In this paper, we propose a meta-heuristic approach based on genetic algorithm and local search to solve the MWEDS problem. Therefore, the proposed method is considered as a memetic search algorithm which is called Memetic Algorithm with filtering scheme for minimum weighted edge dominating set, and called shortly (MAFS). In the MAFS method, three new fitness functions are invoked to effectively measure the solution qualities. The search process in the proposed method uses intensification scheme, called "filtering", beside the main genetic search operations in order to achieve faster performance. The experimental results proves that the proposed method is promising in solving the MWEDS problem.**

Keywords: Minimum weight edge dominating set, Graph theory, Genetic algorithm, Memetic algorithm, Local search.

I. Introduction

The Minimum Edge Dominating Set (MEDS) is a subset of edges of minimum cardinality, where each edge is be in the edge dominating set, or adjacent to some edges in the edge dominating set [11], [12], [24]. The weighted version of MEDS seeks to find an edge dominating set with a minimum total weight [9]. The MEDS problem is a hard combinatorial problem, classified as NP-hard [24], and in general cannot be solved exactly in polynomial time. The MEDS problem is one of the fundamental covering problems in graphs; edge cover, vertex cover, dominating set and edge dominating set. These domination problems in graphs have been subject of many studies in graph theory, and have many applications in operations research, resource allocation and network routing, as well as in coding theory [4], [11], [12], [25].

There are many algorithms proposed for solving MWEDS. Although these algorithms guarantee the optimality of the solutions they find, they may fail to give a solution within reasonable time for large instances. As the size of the problem increases, these methods become futile. Meta-heuristics are powerful search methods which can be efficiently in providing satisfactory solutions to large and complex problems such as vertex cover [20], dominating set [14] and edge coloring [16] in a reasonable time. However, up to the authors' knowledge, there are no studies up to day used meta-heuristic techniques for solving the MWEDS problem.

Genetic Algorithms (GAs) are the most popular meta-heuristic algorithms that have been employed in wide variety of problems [3]. Actually, GAs are able to incorporate other techniques within its framework to produce a hybrid method that brings more promising one. One direction of such hybridization is to use local search which can accelerate the search process in a pure GA. This modification yields another search approach which is called the Memetic Algorithm (MA) [17].

Several meta-heuristic methods have been developed to solve different problems in graph theory and combinatorial optimization [2], [19]. However, the number of contributions that deal with the graph domination problems is very limited. In this paper, we propose a memetic algorithm with filtering scheme for finding the minimum edge dominating set, called shortly MAFS. It uses a 0-1 variable representation of solutions in searching for the MWEDS, and invokes three new fitness functions to measure the solution qualities. Intensification search and filtering schemes are used beside local search in order to enhance the performance of the MAMEDS method.

The paper is organized as follows. The next section gives a brief description about the MWEDS problem as preliminaries needed throughout the paper, and highlights the related works in solving the considered problem. Section 3 describes the proposed method steps in details. Section 4 reports numerical experiments and results. Finally, the conclusions make up Section 5.

II. Problem Formulation and Related Works

Given an undirected weighted graph $G = (V, E, W)$, without loops and multiple edges, where V is the set of nodes (or vertices), E the set of edges, and W is the set of positive edge weights represented by variables $w_1, w_2, \ldots, w_m$, (where each w_i corresponds to an edge $i = (u, v) \in E$). An edge (u, v) of G is said to dominate itself and any edge adjacent to it in G. An edge dominating set (EDS) is a set of edges which is collectively dominate all the other edges in the graph G. The Minimum Weight Edge Dominating Set (MWEDS) problem seeks to find an edge dominating set EDS of minimum total weight $\Sigma_{e \in D} w(e)$.

The edge dominating set problem is a basic problem introduced in Garey and Johnson's work [10] on NP-completeness. Yannakakis and Gavril [24] proved that the edge dominating set problem is NP-hard even in planar or bipartite graphs

of maximum degree 3. Although the EDS has important application in areas such as telephone switching networks, a very little work was known about the weighted version of the problem.

For the EDS problem, Randerath and Schiermeyer [8] presented the first exact algorithm of time complexity $O(1.4423^n)$ algorithm and Fomin et al. [6] improved this to $O(1.4082^n)$. Rooij and Bodlaender [21] got an $O(1.3226^n)$ algorithm by using the "measure and conquer" method, which was further improved to $O(1.3160^n)$ [23], where n is the number of vertices. From the point of approximation algorithm, the best known result was proposed in [18], which gave a 2-approximation algorithm for WEDS problem. Recently, parameterized computation theory was applied to solve the EDS and WEDS problems. Fernau [5] presented parameterized algorithms of time complexity $O(2.62^k)$ for EDS and WEDS problem respectively. The above result was further reduced by Fomin [7], which gave a parameterized algorithm of time $O(2.4181^k)$. For the first time Wang [22] presented an enumeration algorithm of time complexity $O(5.6^{2k}k^4z^2+4^{2k}nk^3z)$ for WEDS problem. Although these algorithms provide the optimal solution, they are too slow on graphs with few hundreds of nodes. Therefore, when deals with a very large graphs, these algorithm become impractical. This motivates us to consider meat-heuristics to design more efficient algorithm to solve the MWEDS problem.

III. Proposed Method

In this section, we describe the components of the MAFS method, and then state its formal algorithm at the end of this section. The MAFS method is an evolutionary algorithm, therefore, we first start by describing the solution representation and the fitness function. Then, the genetic operators; selection, crossover and mutation are defined. The main memetic search element, local search, is stated after that. Finally, our intensification schemes are explained.

A. Graph Representation

The graph represented as $n_V \times n_V$ adjacency matrix A, where n_V is the number of vertices in the graph. The non-diagonal entry $a_{ij} = w_e$, where w_e is an integer weight associated with each edge e connected the vertex i to vertex j. Form an adjacency matrix we create edges matrix E_m which include all edges in the graph. Edge matrix dimension is $n_E \times 3$, where n_E is the number of edges in the graph. The first two columns are the vertex numbers which represent the endpoints of edges and the third columns represent the weights of each edge in the graph.

B. Solution Representation

A solution s will be represented as a bit vector with size equal to the number of edges in the graph. Therefore, s is equal to $(s_1, s_2, \ldots, s_{n_E})$, as shown in Figure 1. The subscript numbers $1, 2, \ldots, n_E$, are related to the corresponding edges in E_m. If a component s_i of s, $i = 1, \ldots, n_E$, has the value 1, then the edge represented by the i-th row in E_m is contained in the edge subset represented by solution s. Otherwise, the solution s does not contain the i-th edge.

s_1	s_2	s_3	$\cdots$	s_{n_E}

Fig. 1. Solution Representation

C. Fitness Function

Fitness function *fit* is a function designed to measures the quality of a solution which plays a major role in the selection process. The main idea in designing the fitness function is that better solutions will have a higher fitness function value than worse one. Three fitness functions are invoked to effectively measure the solution qualities.

$$fit_1(s) = \rho_d + \frac{1}{sum_w(s) \times n_E}, \tag{1}$$

$$fit_2(s) = \rho_d + (1 - \sqrt[\kappa]{\frac{sum_w(s)}{T_{sum}}}), \tag{2}$$

$$fit_3(s) = \alpha\rho_d + (1-\alpha)(1 - \sqrt[\kappa]{\frac{sum_w(s)}{T_{sum}(s)}}), \tag{3}$$

where $0 \leq \alpha \leq 1$, $\kappa > 1$ is an integer, and ρ_d, $T_{sum}(s)$ and $sum_w(s)$ are calculated by

$$\rho_d = \frac{n_D}{n_E},$$

$$T_{sum}(s) = \sum_{e \in E} w(e),$$

$$sum_w(s) = \sum_{e \in D} w(e),$$

where n_D is the number of edges dominated by the subset of edges D represented by the solution s and n_E the number of edges in the graph. All three fitness function consist of two parts, the first part n_D/n_E, reflects the size of domination on G by s. If s represents an edge dominating set, then this part is equal to 1. On the other hand, the second part distinguishes between solutions that have the same values of the first part based on the sum of weights associated with each edge contained in each of them. It is worthwhile to mention that the second term is designed to make $fit(s_1) < fit(s_2)$ in only two cases:

- $x_1 < x_2$, where x_1 and x_2 are the numbers of edges covered by s_1 and s_2 respectively, or
- $x_1 = x_2$ and $sum_w(s1) > sum_w(s2)$.

The parameter κ is set equal to 4 to highly distinguish between solutions that have the same domination number.

D. Genetic Operators

The parent selection mechanism first produces an intermediate population, say P' from the initial population P: $P' \subseteq P$ as in the canonical GA. For each generation, P' has the same size as P but an individual can be present in P' more than once. The individuals in P are ranked with their fitness function values based on the *linear ranking selection mechanism* [1], [13]. Indeed, individuals in P' are copies of individuals in P depending on their fitness ranking: the higher fitness an individual has, the more the probability that it will be copied is. This process is repeated until P' is full while an already chosen individual is not removed from P.

The crossover operation has an exploration tendency, and therefore it is not applied to all parents. First, for each individual in the intermediate population P', the crossover operation chooses a random number from the interval $(0, 1)$. If the chosen number is less than the crossover probability $p_c \in (0, 1)$, the individual is added to the parent pool. After that, two parents from the parent pool are randomly selected and mated to produce two children c_1 and c_2, which are then replacing their parents in P'. These procedures are repeated until all selected parents are mated. The standard one-point crossover [15] is used in MAFS to compute children.

For each gene each in all individuals in the intermediate population P', a random number from the interval $(0, 1)$ is associated. If the associated number is less than the mutation probability p_m, then the individual is mutated using the standard uniform mutation operation [15].

E. Local Search

In *LocalSearch* mechanism, we add or delete some edges to improve the best solution s^{best} found so far, and this process is repeated n_l times. The formal description of this mechanism is shown in Procedure 1.

Procedure 1: (LocalSearch)

1) *Set a suitable value to n_l.*
2) *Repeat the following Steps (2-6) n_l times.*
3) *Set $\tilde{s}^{best}$ = s^{best}.*
4) *If $\rho_d \geq 1$, select a component $\tilde{s_i}^{best}$ with value 1. This selection is randomly and proportional to the weight of its corresponding edge. Set $\tilde{s_i}^{best}$ = 1- $\tilde{s_i}^{best}$.*
5) *If $\rho_d < 1$, select a component $\tilde{s_i}^{best}$ with value 0. This selection is randomly and inversely proportional to the weight of its corresponding edge. Set $\tilde{s_i}^{best}$ = 1- $\tilde{s_i}^{best}$.*
6) *If fit($\tilde{s}^{best}$)> fit(s^{best}), set s^{best} = $\tilde{s}^{best}$.*

In our numerical experiments, the number n_l of local search iterations is set equal to $0.1 \times n_E$ in order to save computational costs.

F. Intensification Schemes

The intensification mechanism which called "*Filtering*" is used in MAFS to reduce the cost of the solution computed. This mechanism basically checks if an edge contained in s^{best} can be removed without losing the coverage.

Procedure 2: (Filtering)

1) *If $\rho_d < 1$, return.*
2) *Compute the set $X = \{x_1, ..., x_{n_E}\}$ of all positions of value one in s^{best}.*
3) *Repeat the following Steps (4-5) for $j = 1, ..., n_E$.*
4) *Set s^{best}_{xj} = 0, and compute the new fitness value.*
5) *If the fitness value is increased, update s^{best} .*

G. MAFS Algorithm

MAFS starts with an initial population of chromosomes P_0 generated randomly. Each chromosome represents a trial solution to the MWEDS problem. During each generation, the quality of each chromosome in the population is evaluated by using three fitness functions (see Equations 1,2 and 3). MAFS applies Procedure 1 to improve the best solution. In each generation, the population is updated through genetic operators. Linear ranking selection algorithm uses to select parents for standard one-point crossover and uniform mutation to generate members of the new population [14]. MAFS invokes Local Search Procedure to update the current population. If a certain number of consecutive generations without improvement is achieved, MAFS invokes Procedure 2 to improve the best edge dominating set s^{best} obtained so far, if it exists. The search will be terminated if the number of generations exceeds g_{max}, or the number of consecutive generations without improvement exceeds a pre-specified number.

Algorithm 3: (MAFS)

1) **Initialization**. Set values of P_{size}, g_{max}. Set the crossover and mutation probabilities $p_c \in (0, 1)$ and $p_m \in (0, 1)$, respectively. Set *WEDS* to be an empty set. Generate an initial population P_0 of size P_{size}.
2) **Local Search**. Evaluate the fitness function of all chromosomes in P_0 by using the Equations 1, 2 or 3, and then apply Procedures 1 to improve the best trial solution in P_0. Set the generation counter $t := 0$.
3) **Parent Selection**. Select an intermediate population $\acute{P}_t$ from the current population P_t using the linear ranking selection.
4) **Crossover.** Apply the standard one-point crossover to chromosomes in $\acute{P}_t$, and update $\acute{P}_t$.
5) **Mutation**. Apply the standard uniform mutation to chromosomes in $\acute{P}_t$, and update $\acute{P}_t$.
6) **Survival Selection**. Evaluate the fitness function of all generated children in the updated $\acute{P}_t$, and set P_t+1 = $\acute{P}_t$. If the best solution in P_t+1 is worse than the best solution in $\acute{P}_t$, then replace the worst solution in $\acute{P}_t$+1 by the best solution in $\acute{P}_t$.
7) **Local Search**. Apply Procedure 1 to improve the s^{best}, update *WEDS*.
8) **Filtering**. If s^{best} represents a weight edge dominating set, then apply Procedure 2 to improve it, update *WEDS*.
9) **Stopping Condition**. If $t > g_{max}$, then terminate. Otherwise, set $t := t + 1$, and go to Step 3.

IV. Numerical Experiments

The MAFS algorithm was programmed using MATLAB. In this experimental section, we technically discuss the

implementation of the MAFS code as well as its results. This section also shows how the test graphs used in the numerical simulations are generated.

A. Graph Generation

In order to measure the performance of MAFS we apply it on number of graphs with different sizes. The previous works in solving MWEDS did not implemented for special types of graphs. Thus, the graphs which we used in our experiments are randomly generated with a known edge domination number $\grave{\gamma}(G)$ and optimal total weight op_w. The following algorithm describe how these graphs are constructed.

Algorithm 4: (Graph Generation)

1) Set the maximum number of edges $max_E = n_V \times (n_V - 1)/2$, and the number of edges $n_E = max_E \times d$, where d is the density of edges in the graph which is set to be in $(0, 1)$, and n_V is the number of vertices.
2) Divide the vertices into two groups:
   - V_{ED} with size equal to $\grave{\gamma}(G) \times 2$, and has vertices incident to dominant edges. Therefore, each pair of them is connected.
   - V_E with size equal to $n_V - (\grave{\gamma}(G) \times 2)$, and has vertices not incident to dominant edges.
3) Add edges to connect the graph vertices to reach the edge density d. This edge adding process should satisfy the following condition in order to maintain the edge domination number equal to $\grave{\gamma}(G)$.
   - No edge connects two vertices belong to different pairs in V_{ED}.
4) Set the weights w_e randomly for each edge in G such that $0 < w_e \leq l_1$ for each dominant edge and $l_1 < w_e \leq l_2$ for the remaining edges.

In our numerical experiments, the parameters l_1 and l_2 is set equal to $\grave{\gamma}(G)$ and n_E, respectively.

MAFS was applied to 15 instances of MWEDS problems created from the five graphs G1-G5, see Table I. These three graphs generated randomly with a number n_V of vertices and different number n_E of edges depending on the density number d for each instance. For each problem instance, the edge domination number $\grave{\gamma}(G)$ and the optimal total weight op_w was known and the code was run 10 times.

B. Parameter Setting

Table II summarizes all parameters setting used in MAFS with their assigned values. These chosen values are based on our numerical experiments.

C. Comparison Results

In this section, we study the performance comparison of the proposed MAFS with three fitness functions that we introduce in Equations 1, 2 and 3. We have two comparison results, the first comparison of MAFS with (fit_1) against MAFS with (fit_2), and the results of this comparison are reported in Table III. The second performance comparison of MAFS with (fit_2) against MAFS with (fit_3), and the results of this comparison are reported in Table IV. To measure the performance of each method, two quantities are used in the comparisons which are computed as follows.

TABLE I. TEST PROBLEMS

Test graphs	n_V	n_E	d	$\grave{\gamma}(G)$	op_w
$G^{20}_{0.1}$	20	19	0.1	4	4
$G^{20}_{0.3}$	20	57	0.3	4	4
$G^{20}_{0.5}$	20	95	0.5	4	4
$G^{30}_{0.1}$	30	44	0.1	8	8
$G^{30}_{0.3}$	30	131	0.3	8	8
$G^{30}_{0.5}$	30	218	0.5	8	8
$G^{50}_{0.1}$	50	123	0.1	15	15
$G^{50}_{0.3}$	50	368	0.3	15	15
$G^{50}_{0.5}$	50	613	0.5	15	15
$G^{100}_{0.1}$	100	495	0.1	28	28
$G^{100}_{0.3}$	100	1485	0.3	28	28
$G^{100}_{0.5}$	100	2475	0.5	28	28
$G^{200}_{0.1}$	200	1990	0.1	46	46
$G^{200}_{0.3}$	200	5970	0.3	46	46
$G^{200}_{0.5}$	200	9950	0.5	46	46

TABLE II. MAMEDS PARAMETER SETTING

Parameter	Definition	Value
P_{size}	Population size	100
p_c	Crossover probability	0.8
p_m	Mutation probability	0.01
n_{EDS}	Max number of the best weight edge dominating sets used to update WEDS	10
g_{max}	Max number of generations	100

1) *Average Number ($Ave.$). This measure gives the average of the optimal solution values found in the independent runs.*
2) *Rate Number ($rate$). The rate shows how many times MAFS acquires an optimal solution op_w.*

1) Performance Comparison of MAFS with (fit_1) and (fit_2): The results of this comparison are reported in Table III. The results show that MAFS with (fit_1) could not acquire the optimal total weight op_w for all instances of the MWEDS problem especially when the number of edges increased proportionally with the graph size. MAFS with (fit_2) achieve significant improvement in the average results and in acquiring the optimal total weight op_w for all instances. However it has a low rate $(rate)$ for instances with large number of edges.

2) Performance Comparison of MAFS with (fit_2) and (fit_3): In this comparison, we compared MAFS with (fit_2) against MAFS with (fit_3). The results of this comparison are reported in Table IV. To achieve the best performance of the MAFS, the fit_2 was moderated by adding weights α, and $(1 - \alpha)$ to get a new fitness function fit_3 in 3. The weight parameters α is set equal to 0.4, which is used to efficiently trade-off between the trail solutions. The comparison results confirm a superior performance of MAFS with fit_3 in both terms $(Ave.)$ and $(rate)$ against the other two fitness functions.

In Table V the instances generated by algorithm 4 with modifications in the role of edge weights such that the dominant edges assigned weights w_e from $\{1, 2, ..., \grave{\gamma}(G)\}$, and for the remaining edges the set $\{\grave{\gamma}(G) + 1, \grave{\gamma}(G) + 2, \ldots, \grave{\gamma}(G) + 30\}$. MAFS with $fit3$ applied for these instances. The results show that when the dominant edges have different weights

TABLE III. RESULTS OF MAFS ON G1-G5 USING TWO FITNESS FUNCTIONS fit_1 AND fit_2

Graph no	n_E	MAFS with fit_1			MWEDS with fit_2		
		op_w	$Ave.$	$rate$	op_w	$Ave.$	$rate$
$G_{0.1}^{20}$	19	4	4	10	4	4	10
$G_{0.3}^{20}$	57	4	4	10	4	4	10
$G_{0.5}^{20}$	95	4	4	10	4	4	10
$G_{0.1}^{30}$	44	8	18.5	3	8	10.8	9
$G_{0.3}^{30}$	131	8	36.9	4	8	9.2	8
$G_{0.5}^{30}$	218	8	40.2	1	8	9.2	8
$G_{0.1}^{50}$	123	15	146	3	15	25.2	6
$G_{0.3}^{50}$	368	15	180	0	15	33.9	3
$G_{0.5}^{50}$	613	15	197	0	15	32.2	4
$G_{0.1}^{100}$	495	28	169	1	28	42.9	4
$G_{0.3}^{100}$	1485	28	187	1	28	47.2	4
$G_{0.5}^{100}$	2475	28	210.9	0	28	50	3
$G_{0.1}^{200}$	1990	46	270	0	46	115	1
$G_{0.3}^{200}$	5970	46	320	0	46	132.2	2
$G_{0.5}^{200}$	9950	46	536	0	46	197	2

TABLE IV. RESULTS OF MAFS ON G1-G5 USING TWO FITNESS FUNCTIONS fit_2 AND fit_3

Graph no	n_E	MAFS with fit_2			MAFS with fit_3		
		op_w	$Ave.$	$rate$	op_w	$Ave.$	$rate$
$G_{0.1}^{20}$	19	4	4	10	4	4	10
$G_{0.3}^{20}$	57	4	4	10	4	4	10
$G_{0.5}^{20}$	95	4	4	10	4	4	10
$G_{0.1}^{30}$	44	8	10.8	9	8	8	10
$G_{0.3}^{30}$	131	8	9.2	8	8	8	10
$G_{0.5}^{30}$	218	8	9.2	8	8	8	10
$G_{0.1}^{50}$	123	15	25.2	6	15	15	10
$G_{0.3}^{50}$	368	15	33.9	3	15	15	10
$G_{0.5}^{50}$	613	15	32.2	4	15	15	10
$G_{0.1}^{100}$	495	28	42.9	4	28	28	10
$G_{0.3}^{100}$	1485	28	47.2	4	28	28	10
$G_{0.5}^{100}$	2475	28	50	3	28	28	10
$G_{0.1}^{200}$	1990	46	115	1	46	46	10
$G_{0.3}^{200}$	5970	46	132.2	2	46	46	10
$G_{0.5}^{200}$	9950	46	197	2	46	46	10

and near of that in non dominant edges the solutions will be more difficult to acquired in every time. Moreover, the MAFS method exhibits very promising performance to obtain MWEDS of graphs.

V. CONCLUSION

The minimum weight edge dominating set problem in graph theory has been studied in this paper. We proposed a memetic-based method to solve this problem called Memetic Algorithm with Filtering Scheme (MAFS). Intensification scheme used beside the genetic and local search methodologies in order to achieve better performance. Three new fitness functions invoked to maximize the performance of the proposed method. These fitness functions consider different ways to balance between two objectives; edge dominating and weight minimizing. Specifically, two of these fitness functions use absolute additions of valued functions that measure the considered objectives while the third one uses a weighted addition way. Numerical experiments of MAFS using the three fitness functions on various test graphs show that the MAFS with a weighted fitness function outperform the MAFS with the other two fitness functions. In addition, the proposed method show very promising performance to obtain minimum weighted edge dominating sets for different graphs used in the numerical experiments.

TABLE V. RESULTS OF MAFS ON G1-G5 USING FITNESS FUNCTIONS fit_3

Graph no	n_E	op_w	$Ave.$	$rate$
$G_{0.1}^{20}$	19	10	10	10
$G_{0.3}^{20}$	57	10	10	10
$G_{0.5}^{20}$	95	10	10	10
$G_{0.1}^{30}$	44	36	36	10
$G_{0.3}^{30}$	131	36	36.8	9
$G_{0.5}^{30}$	218	36	36.5	7
$G_{0.1}^{50}$	123	120	121.8	8
$G_{0.3}^{50}$	368	120	122.5	7
$G_{0.5}^{50}$	613	120	127.8	8
$G_{0.1}^{100}$	495	406	420	8
$G_{0.3}^{100}$	1485	406	427.8	8
$G_{0.5}^{100}$	2475	406	433.8	7
$G_{0.1}^{200}$	1990	1081	1150	7
$G_{0.3}^{200}$	5970	1081	1220	8
$G_{0.5}^{200}$	9950	1081	1300	6

REFERENCES

[1] James E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[2] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[3] Lance D Chambers. *Practical handbook of genetic algorithms: complex coding systems*, volume 3. CRC press, 2010.

[4] Zhifang Feng, Wensheng Li, Huaming Xing, and Qiongyu Ma. On the upper bounds of local inverse signed edge domination numbers in graphs. In *Information Computing and Applications*, pages 368–372. Springer, 2012.

[5] Henning Fernau. Edge dominating set: Efficient enumeration-based exact algorithms. In *Parameterized and Exact Computation*, pages 142–153. Springer, 2006.

[6] Fedor V Fomin, Serge Gaspers, Saket Saurabh, and Alexey A Stepanov. On two techniques of combining branching and treewidth. *Algorithmica*, 54(2):181–207, 2009.

[7] Fedor V Fomin, Serge Gaspers, Saket Saurabh, and Alexey A Stepanov. On two techniques of combining branching and treewidth. *Algorithmica*, 54(2):181–207, 2009.

[8] Fedor V Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and conquer: domination–a case study. In *Automata, Languages and Programming*, pages 191–203. Springer, 2005.

[9] Toshihiro Fujito and Hiroshi Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discrete Applied Mathematics*, 118(3):199–207, 2002.

[10] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman New York, 1979.

[11] Teresa W Haynes, Stephen T Hedetniemi, and Peter J Slater. *Domination in graphs: advanced topics*, volume 40. Marcel Dekker New York, 1998.

[12] Teresa W Haynes, Stephen T Hedetniemi, and Peter JB Slater. *Fundamentals of Domination in Graphs*, volume 208. CRC PressI Llc, 1998.

[13] Abdel-Rahman Hedar and Masao Fukushima. Minimizing multimodal functions by simplex coding genetic algorithm. *Optimization Methods and Software*, 18(3):265–282, 2003.

[14] Abdel-Rahman Hedar and Rashad Ismail. Hybrid genetic algorithm for minimum dominating set problem. In *Computational Science and Its Applications–ICCSA 2010*, pages 457–467. Springer, 2010.

[15] Francisco Herrera, Manuel Lozano, and Jose L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial intelligence review*, 12(4):265–319, 1998.

[16] Tiago Januario, Sebastián Urrutia, Belo Horizonte, and Minas Gerais-Brazil. An edge coloring heuristic based on vizings theorem. In *Sejam bem-vindos aos pr-anais XVI CLAIO*. Conference Proceedings, 2012.

[17] Pablo Moscato, Carlos Cotta, and Alexandre Mendes. Memetic algorithms. In *New optimization techniques in engineering*, pages 53–85. Springer, 2004.

[18] Ojas Parekh. *Polyhedral techniques for graphic covering problems*. PhD thesis, Department of Mathematical Science, Carnegie Mellon University, 2002.

[19] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. Wiley, 2009.

[20] Onur Ugurlu. New heuristic algorithm for unweighted minimum vertex cover. In *IV International Conference "Problems of Cybernetics And Informatics"*. Conference Proceedings, 2012.

[21] Johan MM Van Rooij and Hans L Bodlaender. Exact algorithms for edge domination. In *Parameterized and Exact Computation*, pages 214–225. Springer, 2008.

[22] Jianxin Wang, Beiwei Chen, Qilong Feng, and Jianer Chen. An efficient fixed-parameter enumeration algorithm for weighted edge dominating set. In *Frontiers in Algorithmics*, pages 237–250. Springer, 2009.

[23] Mingyu Xiao and Hiroshi Nagamochi. A refined exact algorithm for edge dominating set. In *Theory and Applications of Models of Computation*, pages 360–372. Springer, 2012.

[24] Mihalis Yannakakis and Fanica Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.

[25] Yancai Zhao, Lianying Miao, Haichao Wang, and Wenyao Song. Maximal matching and edge domination in complete multipartite graphs. *International Journal of Computer Mathematics*, (to appear), 2013.