Optimization Letters manuscript No.
(will be inserted by the editor)

Integer Programming Formulations for the Minimum Weighted Maximal Matching Problem

Z. Caner Taşkın · Tınaz Ekim

Received: date / Accepted: date

Abstract Given an undirected graph, the problem of finding a maximal matching that has minimum total weight is NP-hard. This problem has been studied extensively from a graph theoretical point of view. Most of the existing literature considers the problem in some restricted classes of graphs and give polynomial time exact or approximation algorithms. On the contrary, we consider the problem on general graphs and approach it from an optimization point of view. In this paper, we develop integer programming formulations for the minimum weighted maximal matching problem and analyze their efficacy on randomly generated graphs. We also compare solutions found by a greedy approximation algorithm, which is based on the literature, against optimal solutions. Our results show that our integer programming formulations are able to solve medium size instances to optimality and suggest further research for improvement.

Keywords minimum maximal matching · edge dominating set · integer programming

Z. Caner Taşkın's research was partially supported by Boğaziçi University Research Fund Grant No: 5028 and the research of Tınaz Ekim was supported by Grant 108M616 of CNRS-TÜBİTAK joint research project, whose support is greatly acknowledged.

Z. C. Taşkın
Department of Industrial Engineering, Boğaziçi University
34342 Bebek, İstanbul, Turkey
E-mail: caner.taskin@boun.edu.tr

T. Ekim
Department of Industrial Engineering, Boğaziçi University
34342 Bebek, İstanbul, Turkey
E-mail: tinaz.ekim@boun.edu.tr

1 Introduction and Literature Survey

Given a graph, a *matching* is a set of edges that are pairwise non-adjacent. A matching is said to be *maximal* if no other edge can be added to it while keeping the property of being a matching. We say that an edge in a matching *dominates* itself and all edges adjacent to it. Furthermore, nodes adjacent to edges of a matching are said to be *saturated* by this matching. Note that in a maximal matching of a graph $G = (N, E)$ every edge in E is necessarily dominated. The problem of finding a maximal matching having minimum cardinality is called Minimum Maximal Matching (MMM). In the presence of edge weights, the objective becomes minimization of the sum of the weights in a maximal matching, and the related problem is called Minimum Weighted Maximal Matching (MWMM).

MMM has been studied extensively due to its various applications. Yannakakis and Gavril [16] describe an application related to a telephone switching network built to route phone calls from incoming lines to outgoing trunks. The worst case behavior of such a network (minimum number of calls routed when the system is saturated) can be evaluated by a minimum maximal matching. Another application appears in the theory of stable marriages; MMM gives an upper bound on the number of unmatched candidates in a stable marriage, which is naturally desired to be as small as possible [6].

A similar problem, called minimum edge dominating set (EDS), can be stated as follows: given a graph $G = (N, E)$, find a set of edges having minimum cardinality that dominates all edges in E. It is well known that the size of a minimum maximal matching is equal to the size of a minimum edge dominating set, and given a minimum edge dominating set, it is easy to find a minimum maximal matching [16]. However, as pointed out in [8], this close relationship is lost if edge weights are introduced.

Unlike the problem of finding a maximum matching in a given graph, which is known to be polynomially solvable by Edmond's augmenting path algorithm [7], MMM is NP-hard in several classes of graphs including bipartite or planar graphs with maximum degree 3 [16], planar bipartite graphs, planar cubic graphs [9] and k-regular bipartite graphs for any fixed $k \geq 3$ [6]. Similarly, while the maximum flow problem is polynomially solvable, the minimum maximal flow problem is NP-hard [14]. In the literature, MMM is handled mainly with graph theoretical tools. For instance, polynomially solvable cases for MMM can be found in [11] for trees, in [10] for block graphs, in [12] for series-parallel graphs, and in [15] for bipartite permutation graphs and cotrianglated graphs.

There are also some results on MMM from an approximation point of view. It is well known that in general, a maximum matching cannot be larger than twice the size of a minimum maximal matching since one edge in a minimum maximal matching can cover at most two edges of a maximum matching. Hence, any maximal matching gives a 2-approximation for MMM. Surprisingly, no better approximation is known in general. Moreover, a 2-approximation can also be obtained for the weighted version of EDS [8]. In [5], it is shown that

MMM cannot be approximated within factor $\frac{7}{6}$ unless P=NP, which leaves space for a possible improvement between $\frac{7}{6}$ and 2. More recently, Cardinal et al. [4] give an approximation ratio strictly less than 2 for dense graphs. Slight improvements on these results can be obtained by making certain assumptions on the minimum degree and the density of the graphs ([3] and [5]).

To the best of our knowledge, MMM and MWMM have not been addressed from an optimization point of view in the literature. Integer programming formulations are used to derive approximation algorithms for weighted EDS [8] and for a generalized version of weighted EDS, where capacities and demands are introduced on edges [1]. However, these methods are not applicable to MWMM; as already mentioned, the presence of weights breaks the close relationship between the two problems [8].

In this paper, we first give a natural integer programming formulation of MWMM (Model 1). We then improve this formulation by adding new variables to indicate saturated nodes in a solution (Model 2), which we then enhance by re-formulating some constraints. This new formulation (Model 3) also allows the derivation of some valid inequalities. We evaluate computational efficacy of Model 1 and Model 3 on randomly generated graphs for both weighted and unweighted edges. We also compare the results obtained by a greedy algorithm for MWMM with optimal solutions. Our experiments show that Model 3 provides satisfactory results solving almost all instances with up to 170 nodes to optimality in a reasonable amount of computational time. Furthermore, the greedy algorithm gives good quality results for unweighted instances whereas the quality considerably deteriorates when weights are introduced.

2 An Integer Programming Approach

Let $G = (N, E)$ be an undirected graph and c_{ij} denote the weight associated with edge $(i, j) \in E$. We define binary variable $x_{ij} = 1$ if edge $(i, j) \in E$ is selected in an optimal solution, and 0 otherwise. Let $A(i)$ denote the set of nodes that are adjacent to node $i \in N$. MWMM can be formulated as:

$$\text{(Model 1)} \quad \text{Minimize} \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to:} \sum_{j\in A(i)} x_{ij} \leq 1 \quad \forall i \in N \tag{2}$$

$$\sum_{\substack{k\in A(i)\\k\neq j}} x_{ik} + \sum_{\substack{k\in A(j)\\k\neq i}} x_{kj} + x_{ij} \geq 1 \quad \forall (i,j) \in E \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E. \tag{4}$$

The objective function (1) minimizes the total weight corresponding to the selected edges. Constraints (2) enforce that at most one edge emanating from each node can be selected (hence guaranteeing that the solution is a matching). Constraints (3) ensure that each edge is either selected or is adjacent to a selected edge, which guarantees that the generated matching is maximal. Finally, Constraints (4) enforce binary restrictions on the x-variables.

3 An Improved Formulation

Our preliminary computational tests showed that Model 1 can only solve problems on small graphs to optimality due to the weak lower bound induced by the linear programming relaxation (see Tables 1 and 2). In our tests, we observed that integer programming solvers failed to generate effective cuts to improve the lower bound. Furthermore, no special structure that can be exploited to generate valid inequalities is apparent in Model 1. In order to alleviate these difficulties, let us define binary variable $y_i = 1$ if node $i \in N$ is saturated in the matching, that is $\sum_{j \in A(i)} x_{ij} = 1$. Using these additional variables, we can formulate the problem as:

$$
\begin{aligned}
\text{(Model 2) Minimize} \quad & \sum_{(i,j) \in E} c_{ij} x_{ij} && (5)\\
\text{subject to:} \quad & \sum_{j \in A(i)} x_{ij} \le 1 \quad \forall i \in N && (6)\\
& y_i + y_j \ge 1 \quad \forall (i,j) \in E && (7)\\
& x_{ij} \le y_i \quad \forall i \in N, j \in A(i) && (8)\\
& y_i \le \sum_{j \in A(i)} x_{ij} \quad \forall i \in N && (9)\\
& x_{ij} \in \{0,1\} \quad \forall (i,j) \in E && (10)\\
& 0 \le y_i \le 1 \quad \forall i \in N. && (11)
\end{aligned}
$$

Similar to Model 1, the objective function (5) minimizes the total weight corresponding to the selected edges and Constraints (6) ensure that the generated solution is a matching. Constraints (7) enforce the condition that at least one end point of each edge has to be saturated in a maximal matching (otherwise that edge can be added to the matching, contradicting maximality). Constraints (8) ensure that node i is saturated ($y_i = 1$) if some edge emanating from it is selected. Constraints (9) enforce the reverse condition that if node i is saturated, then at least one of the edges emanating from it has to be selected. Note that if the x-variables are binary-valued, then (8) and (9) will force y-variables to take on binary values. Therefore, we relax y-variables as continuous in (11).

We observe that Constraints (7) can be improved as follows: if $x_{ij} = 1$ for some $(i,j) \in E$, then Constraints (8) imply that $y_i = y_j = 1$. Therefore, (7) can be tightened as $y_i + y_j - x_{ij} \ge 1$. Furthermore, Constraints (6), (8) and (9) imply that $\sum_{j \in A(i)} x_{ij} = y_i$ in an optimal solution. Hence, the problem can equivalently be formulated as:

$$\text{(Model 3)}\quad \text{Minimize} \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{12}$$

$$\text{subject to:} \sum_{j\in A(i)} x_{ij} = y_i \quad \forall i \in N \tag{13}$$

$$y_i + y_j - x_{ij} \geq 1 \quad \forall (i,j) \in E \tag{14}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E \tag{15}$$

$$0 \leq y_i \leq 1 \quad \forall i \in N. \tag{16}$$

Proposition 1 *Model 1 is equivalent to Model 3.*

Proof Let $\hat{x}$ be a solution of Model 1 having objective function value $\hat{z}$. Let $\hat{y}_i = \sum_{j\in A(i)} \hat{x}_{ij}$, $\forall i \in N$. Then $(\hat{x}, \hat{y})$ is a solution of Model 3 having the same objective function value $\hat{z}$. Similarly, if $(\bar{x}, \bar{y})$ is a solution of Model 3 having objective function value $\bar{z}$, then $\bar{x}$ is a feasible solution of Model 1 having the same objective function value $\bar{z}$.

Note that Proposition 1 also holds for the linear programming relaxations of the two models. Therefore, the linear programming bounds of the two models are equal. Both models have the same number of constraints and binary variables. Model 3 has $|N|$ additional (continuous) variables, which increases the total number of variables but also makes the technology matrix more sparse. Furthermore, Constraints (7) and (14) in our revised formulations reveal an embedded vertex cover structure in terms of the y-variables. Therefore, valid inequalities exploiting the vertex cover structure can be devised. Berger et al. [1] observe a similar structure in their formulation of the edge dominating set problem, and propose adding the well-known odd hole inequalities. Let $H \subseteq N$ denote the set of nodes that form an odd hole, that is the set of nodes in H form a cycle having length equal to an odd number. Then, Constraints (17) are valid [1]:

$$\sum_{i\in H} y_i \geq \left\lceil \frac{|H|}{2} \right\rceil \quad \forall H \subseteq N \text{ and odd hole.} \tag{17}$$

Also let $C \subseteq N$ denote a clique in N. Since C has an edge between each node pair that has to be covered as enforced by (14), at most one node in C can be unsaturated (see also [2]) . Therefore, the following is valid:

$$\sum_{i\in C} y_i \geq |C| - 1 \quad \forall C \subseteq N \text{ and clique.} \tag{18}$$

These clique inequalities, although quite natural, do not seem to have been used in integer programming formulations of the vertex cover problem in the literature.

Finally, let $i \in N$ be a node having degree 1, and $j \in N$ be its only neighbor. Since at least one of nodes i and j has to be saturated in a maximal matching due to (14), and since the saturation of node i implies that node j is also saturated by the matching, it follows that node j has to be saturated in any maximal matching. Therefore, the following variable fixing rule is valid:

$$y_j = 1 \quad \forall (i,j) \in E, |A(i)| = 1. \tag{19}$$

Remark 1 The valid inequalities (17)–(19) can also be expressed in terms of the x-variables so that they can be used to tighten Model 1. Using the relationship between the x- and y-variables enforced by (13), the odd hole inequalities (17) can be written as follows

$$2 \sum_{\substack{(i,j)\in E \\ i\in H, j\in H}} x_{ij} + \sum_{\substack{(i,j)\in E \\ i\in H, j\notin H}} x_{ij} + \sum_{\substack{(i,j)\in E \\ i\notin H, j\in H}} x_{ij} \geq \left\lceil \frac{|H|}{2} \right\rceil \quad \forall H \subseteq N \text{ and odd hole.} \tag{20}$$

Similarly, the clique inequalities (18) can be written as

$$2 \sum_{\substack{(i,j)\in E \\ i\in C, j\in C}} x_{ij} + \sum_{\substack{(i,j)\in E \\ i\in C, j\notin C}} x_{ij} + \sum_{\substack{(i,j)\in E \\ i\notin C, j\in C}} x_{ij} \geq |C| - 1 \quad \forall C \subseteq N \text{ and clique,} \tag{21}$$

and our fixing rule (19) can be expressed as

$$\sum_{i'\in A(j)} x_{i'j} = 1 \quad \forall (i,j) \in E, |A(i)| = 1. \tag{22}$$

Note that (22) implies the corresponding (2) for each node that it is written for. While (20)–(22) are equivalent to (17)–(19) with respect to their strengths, it should be noted that they have a large number of nonzero coefficients, resulting in significantly denser cuts.

4 Computational Results

We implemented the formulations discussed in the previous sections using CPLEX 12.2 running on a Windows 7 PC with a 2.13 GHz Intel Core i3 CPU and 4 GB RAM. Our base test data set consists of randomly generated problem instances for which the expected edge density of the graph (measured as $D = \frac{2|E|}{|N|\times(|N|-1)}$) takes values 0.3, 0.5 and 0.7. In generating weighted instances, we first generated random graphs as in the unweighted case, and then assigned an integer weight uniformly distributed between 1 and 10 to each edge. We generated ten problem instances for each problem size, which is determined by the expected edge density and the number of nodes.

We used the default options of CPLEX for solving the integer programming problems. Preliminary computational experience on Model 3 indicated that our variable fixing rule (19) and clique inequalities (18) are very effective at closing the integrality gap but use of the odd hole inequalities (17) is not computationally justified. Therefore, in our experiments we only generated (18) and (19). Specifically, we first enumerated all maximal cliques in the input graph and generated a constraint of type (18) for each maximal clique identified. Note that there is an exponential number of maximal cliques in general; however, for problem sizes that we tested they can be enumerated in a fraction of Model 3's solution time. We then added the generated constraints as "user cuts," which CPLEX checks for violation and automatically adds to tighten the formulation as needed. We enforced (19) by setting the lower bound of the corresponding y_i-variable to 1. We imposed a 1200 seconds time limit (including the cut generation time), past which we halted the execution of CPLEX in all our experiments. Furthermore, we implemented Cardinal et al. [4]'s greedy algorithm for MMM, which is known to have an approximation ratio of 2 (and strictly less than 2 for dense graphs). While the algorithm in [4] is originally given for the unweighted version of the problem, we slightly generalized it to handle the weighted case. Our implementation of the greedy algorithm keeps track of the edges in sets U and S corresponding to unprocessed and selected edges, respectively:

- Step 0: $U \leftarrow E, S \leftarrow \emptyset$.
- Step 1: If $U = \emptyset$, then return S; else continue to Step 2.
- Step 2: Pick an edge (i, j) in U such that

$$\frac{c_{ij}}{\sum_{i'j' \in U \cap \delta_{(i,j)}} c_{i'j'} + c_{ij}}$$

  is minimized, where $\delta_{(i,j)}$ is the set of all edges that are adjacent to (i, j).
- Step 3: Add (i, j) to S, remove $(i, j) \cup \delta_{(i,j)}$ from U, return to Step 1.

This algorithm behaves the same as the algorithm proposed in [4] for the unweighted case, and also handles the weighted case in a similar manner. However, the approximation analysis given in [4] explicitly assumes that all edges have unit weight and therefore the same analysis does not hold for the weighted case.

Our first experiment compares the performances of Model 1, Model 3 and the greedy algorithm on the unweighted case. Our preliminary tests showed that while the solvability of Model 1 is enhanced by the addition of valid inequalities (21) and (22), the corresponding inequalities (18) and (19) are significantly more effective in decreasing the overall solution time of Model 3. Therefore, we only added (18) and (19) to Model 3 in order to quantify the effect of our enhancements over the basic formulation Model 1. Table 1 summarizes the results of Model 1, Model 3 and the greedy algorithm on graphs having edge density $D = 0.3, 0.5$, and 0.7. For each problem size, we report the following statistics calculated over ten random instances: (i) "Solved:" the number of problem instances solved to optimality, (ii) "LP Gap:" the average percentage gap between the linear programming relaxation and the optimal objective function value, (iii) "Gap:" the average final percentage optimality

Table 1 Comparison of Model 1, Model 3 and greedy algorithm on small instances without edge weights

		Model 1				Model 3				Greedy	
D	N	Solved	LP Gap	Gap	Time	Solved	LP Gap	Gap	Time	Gap	Time
0.3	30	10	19.95	-	0.7	10	3.60	-	0.2	15.89	0.0
	40	10	28.43	-	24.0	10	7.10	-	0.4	11.19	0.0
	50	8	31.07	6.81	274.4	10	6.24	-	0.8	11.54	0.0
	60	1	33.73	8.83	1111.0	10	6.41	-	1.9	10.58	0.0
	70	0	35.00	10.02	-	10	6.32	-	2.9	10.25	0.1
0.5	30	10	31.14	-	48.4	10	4.92	-	0.3	9.87	0.0
	40	2	35.17	7.96	723.1	10	4.54	-	0.5	5.69	0.0
	50	0	37.83	10.72	-	10	5.50	-	1.2	6.68	0.0
	60	0	38.59	11.67	-	10	4.36	-	2.8	8.73	0.1
	70	0	40.52	13.97	-	10	5.54	-	5.2	7.37	0.1
0.7	30	10	34.47	-	53.2	10	2.41	-	0.3	8.46	0.0
	40	0	39.17	11.56	-	10	4.12	-	0.6	4.97	0.0
	50	0	40.56	12.36	-	10	4.23	-	1.8	6.80	0.0
	60	0	41.50	12.93	-	10	3.57	-	3.2	6.07	0.1
	70	0	42.71	14.38	-	10	4.21	-	5.2	5.16	0.2

gap for instances that could not be solved within the allowed time limit by the formulations (for the greedy algorithm we report the average percent difference between greedy algorithm's result and the optimal objective function value found by Model 3) , (iv) "Time:" the average amount of time in seconds spent by each algorithm (for Model 1 and Model 3 we report the average on the instances that were solved to optimality within the allowed time limit).

Out of the 150 instances in this data set, CPLEX could solve Model 1 to optimality for only 51 instances within 1200 seconds while being able to solve all instances to optimality for Model 3 within a few seconds. The results show that Model 3 significantly outperforms Model 1. Note that linear programming relaxations of Model 1 and Model 3 are equivalent (Proposition 1). Therefore, the difference between the "LP Gap" values of the two formulations is the result of our clique inequalities and variable fixing rule, which are highly effective in closing the optimality gap. We observe that the linear programming relaxation gets weaker and our inequalities get more effective as the density increases. The greedy algorithm executed very quickly on all problem instances in this data set. While the greedy algorithm could find an optimal solution for only 9 problem instances, it generated solutions having an average optimality gap of 8.62%, which corresponds to choosing 1.88 more edges than an optimal solution. We observe that the performances of Model 1 and Model 3 deteriorate as the number of nodes in the graph and edge density increase. This is expected since the number of binary variables in both formulations increase with the number of edges. On the other hand, note that the quality of the solution found by the greedy algorithm tends to improve as the problem size increases. This can be explained by observing that as the total number of edges increases, the selection decision about each individual edge becomes less critical in the overall solution quality.

Table 2 Comparison of Model 1, Model 3 and greedy algorithm on small instances with edge weights

		Model 1				Model 3				Greedy	
D	N	Solved	LP Gap	Gap	Time	Solved	LP Gap	Gap	Time	Gap	Time
0.3	70	10	15.58	-	2.5	10	4.67	-	0.8	29.76	0.1
	80	10	18.33	-	5.8	10	4.75	-	1.2	30.56	0.1
	90	10	21.43	-	18.6	10	5.07	-	1.9	34.76	0.2
	100	10	21.58	-	45.5	10	4.60	-	2.6	33.24	0.3
	110	4	23.95	6.11	421.1	10	5.66	-	5.8	30.12	0.4
	120	1	25.24	7.53	307.7	10	6.33	-	11.0	29.36	0.6
0.5	70	10	24.41	-	33.1	10	3.65	-	1.1	31.22	0.1
	80	10	25.61	-	220.1	10	4.20	-	2.0	28.32	0.2
	90	1	27.75	6.63	597.0	10	4.12	-	4.0	27.80	0.3
	100	0	30.33	10.26	-	10	4.67	-	6.8	20.86	0.5
	110	0	31.51	13.38	-	10	5.02	-	18.5	25.98	0.7
	120	0	33.54	16.87	-	10	5.91	-	34.9	24.75	1.0
0.7	70	9	28.56	8.81	233.5	10	2.50	-	2.1	28.80	0.2
	80	0	30.79	8.14	-	10	3.02	-	3.9	25.31	0.3
	90	0	33.07	12.15	-	10	3.57	-	7.2	23.98	0.4
	100	0	35.43	15.53	-	10	5.24	-	23.2	18.97	0.7
	110	0	36.64	16.51	-	10	4.93	-	41.1	19.85	1.0
	120	0	37.78	18.53	-	10	5.03	-	90.3	20.37	1.4

Our second experiment compares the three approaches on weighted problem instances. We observe by comparing Tables 1 and 2 that Model 1 is able to solve larger problem instances to optimality compared to the unweighted case. This result is not surprising since it is known that symmetry can make integer programming problems very difficult to solve (see [13]) and adding edge weights eliminates some symmetry by differentiating edges. As before, Model 3 is able to solve all problem instances in this data set within a few seconds of CPU time. We also observe that the quality of the solutions generated by the greedy algorithm is significantly lower than the results for the unweighted case.

Our next set of experiments are aimed at analyzing the performances of Model 3 and greedy algorithm on larger problem instances. Table 3 summarizes the results of tests performed on unweighted graphs. As before, we report the percentage optimality gaps of the linear programming relaxation with and without our clique inequalities and variable fixing rule. We note that observations similar to Table 1 can be made about the performance of Model 3. In particular, we note that the problem becomes harder to solve for CPLEX as D and N increase, which results in a higher number of binary variables in Model 3. Comparison of the optimality gaps calculated for the greedy algorithm on this data set with Table 1 shows that the greedy algorithm tends to get closer to optimality as N and D increase. Note that optimal solutions of some problem instances having $N \geq 170$ are not known. Since we used the best lower bound reported by CPLEX for calculating greedy algorithm's gaps, the gaps reported for these instances possibly overestimate the true gap between the greedy algorithm's solution and the optimal objective function value.

Table 3 Comparison of Model 3 and greedy algorithm on large instances without edge weights

		Model 3					Greedy	
D	N	Solved	LP Gap (No cuts)	LP Gap	Gap	Time	Gap	Time
0.3	150	10	42.40	9.34	-	197.6	6.51	1.4
	160	10	42.94	9.67	-	374.8	5.58	1.8
	170	10	43.03	9.45	-	743.4	5.61	2.3
	180	4	43.54	9.81	2.78	938.8	7.31	2.9
	190	0	44.05	10.21	3.99	-	8.66	3.6
0.5	150	10	45.09	7.28	-	255.8	4.03	2.5
	160	10	45.05	6.86	-	353.3	3.67	3.2
	170	7	45.51	7.33	3.22	753.5	4.84	4.1
	180	1	46.04	7.91	3.08	1191.7	5.79	5.2
	190	0	46.26	8.06	4.51	-	7.72	6.5
0.7	150	10	46.14	5.34	-	558.1	3.30	3.5
	160	10	46.33	5.46	-	901.4	3.84	4.6
	170	2	46.69	5.82	3.57	1024.8	5.68	5.9
	180	0	46.82	5.82	5.23	-	7.71	7.4
	190	0	47.06	6.01	5.83	-	8.27	9.2

Table 4 Comparison of Model 3 and greedy algorithm on large instances with edge weights

		Model 3					Greedy	
D	N	Solved	LP Gap (No cuts)	LP Gap	Gap	Time	Gap	Time
0.3	150	10	28.57	7.07	-	56.4	28.41	1.4
	160	10	30.24	7.68	-	160.0	27.67	1.8
	170	10	31.16	7.78	-	186.6	26.88	2.4
	180	8	33.04	8.73	2.81	501.2	29.33	3.0
	190	5	33.71	9.29	4.36	650.1	27.19	3.7
0.5	150	10	36.79	6.87	-	129.6	20.90	2.5
	160	10	37.59	6.83	-	273.3	20.07	3.3
	170	10	38.67	7.17	-	457.0	17.93	4.2
	180	5	39.35	7.36	2.40	785.9	17.69	5.4
	190	2	40.17	7.77	4.04	1182.4	20.10	6.7
0.7	150	10	40.33	5.30	-	340.1	15.61	3.6
	160	10	41.48	5.45	-	472.0	12.45	4.7
	170	9	41.93	5.68	3.83	860.7	14.87	5.9
	180	3	42.52	5.71	2.57	1023.0	14.07	7.5
	190	0	43.16	5.90	4.53	-	13.90	9.4

Finally, Table 4 summarizes the results obtained on large instances having weighted edges by Model 3 and the greedy algorithm. As before, Model 3 can solve more weighted problem instances to optimality than unweighted instances (112 and 84, respectively). However, unlike Model 1, we observe that the difference between solvable problem sizes is not very significant for Model 3. This can be explained by the fact that our clique inequalities (18) and variable fixing rule (19) are solely based on the topology of the graph, and they do not use any information about edge weights. Comparison of Tables 2 and 4 reveals that quality of the solutions generated by the greedy algorithm tends to improve as N and D increase, which is in alignment with our earlier observations.

5 Conclusions and Future Research

To the best of our knowledge, this paper presents the first integer programming formulations and computational results on MMM and MWMM. Our results constitute a basis for the comparison of the performances of various exact methods, heuristics and approximation algorithms. One potential future research topic is to adjust our formulations to specific graph types. Even though our formulations can be used on general graphs, it may be possible to improve them by deriving bounds and valid inequalities for specific graph types. A particular graph type of interest in the literature is regular graphs, i.e. graphs where all nodes have the same degree. It is known that MMM is NP-hard in regular bipartite graphs but performances of various heuristics and integer programming formulations such as Models 1 and 3 on regular graphs are unknown. More studies are needed to answer whether the structure of regularity makes it easier to solve MWMM, or the high symmetry caused by regularity makes the formulations and heuristics less effective.

References

1. A. Berger, T. Fukunaga, H. Nagamochi, and O. Parekh. Approximability of the capacitated b-edge dominating set problem. *Theoretical Computer Science*, 385:202–213, 2007.
2. I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, 1999.
3. J. Cardinal, M. Labbé, S. Langerman, E. Levy, and H. Mélot. A tight analysis of the maximal matching heuristic. *COCOON, Lecture Notes in Computer Science*, 3595:701–709, 2005.
4. J. Cardinal, S. Langerman, and E. Levy. Improved approximation bounds for edge dominating set in dense graphs. *Theoretical Computer Science*, 410:949–957, 2009.
5. M. Chlebík and J. Chlebíková. Approximation hardness of edge dominating set problems. *Journal of Combinatorial Optimization*, 11(3):279–290, 2006.
6. M. Demange and T. Ekim. Minimum maximal matching is NP-hard in regular bipartite graphs. *TAMC 2008, Lecture Notes in Computer Science*, 4978:364–374, 2008.
7. J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
8. T. Fujito and H. Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discrete Applied Mathematics*, 118:199–207, 2002.
9. J.D. Horton and K. Kilakos. Minimum edge dominating sets. *SIAM Journal on Discrete Mathematics*, 6(3):375–387, 1993.
10. S.F. Hwang and G.J. Chang. The edge domination problem. *Discuss. Math. Graph. Theory*, 15(1):51–57, 1995.
11. S.L. Mitchell and S.T. Hedetniemi. Edge domination in trees. In *Proceedings of the 8th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 489–509, Louisiana State University, Baton Rouge, La., 1977.
12. M.B. Richey and R.G. Parker. Minimum-maximal matching in series-parallel graphs. *European Journal of Operations Research*, 33(1):98–105, 1988.
13. H. D. Sherali and J. C. Smith. Improving discrete model representations via symmetry considerations. *Management Science*, 47:1396–1407, 2001.
14. J. Shi and Y. Yamamoto. A global optimization method for minimum maximal flow problem. *ACTA Mathematica Vietnamica*, 22(1):271–287, 1997.

15. A. Srinivasan, K. Madhukar, P. Nagavamsi, C. Pandu Rangan, and M.-S. Chang. Edge domination on bipartite permutation graphs and cotriangulated graphs. *Information Processing Letters*, 56(3):165–171, 1995.
16. M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38:364–372, 1980.