Decomposition Algorithms for Solving the Minimum Weight Maximal Matching Problem*

Merve Bodur† Tınaz Ekim† Z. Caner Taşkın†

June 19, 2013

Abstract

We investigate the problem of finding a maximal matching that has minimum total weight on a given edge-weighted graph. Although the minimum weight maximal matching problem is NP-hard in general, polynomial time exact or approximation algorithms on several restricted graph classes are given in the literature. In this paper, we propose an exact algorithm for solving several variants of the problem on general graphs. In particular, we develop integer programming formulations for the problem and devise a decomposition algorithm, which is based on a combination of integer programming techniques and combinatorial matching algorithms. Our computational tests on a large suite of randomly generated graphs show that our decomposition approach significantly improves the solvability of the problem compared to the underlying integer programming formulation.

Keywords: minimum maximal matching, vertex cover, mixed integer programming, Benders decomposition, Gallai-Edmonds decomposition.

1 Introduction and Literature Survey

A *matching* on a graph is defined as a set of edges with no common vertex. Vertices that are the end-points of edges of a matching are said to be *saturated* by this matching. Non-saturated vertices are called *exposed* with respect to the matching under consideration. A *maximal matching* is a matching M of a graph G with

*Z. Caner Taşkın's research was partially supported by Boğaziçi University Research Fund Grant No: 6540 and the research of Tınaz Ekim was supported by Grant 108M616 of CNRS-TÜBİTAK joint research project, whose support is greatly acknowledged.

†Department of Industrial Engineering, Boğaziçi University 34342 Bebek, Istanbul, Turkey
Email addresses: {merve.bodur, tinaz.ekim, caner.taskin}@boun.edu.tr,

the property that if any edge not in M is added to M, M is no longer a matching. The minimum maximal matching problem (MMM) seeks a maximal matching that has minimum cardinality [15]. In this paper, we consider a more general version of MMM, where each edge has a weight and the objective is to minimize the total edge weight in a maximal matching. The resulting problem is called minimum weight maximal matching (MWMM) [31]. Throughout our solution procedure, we also introduce the problem of finding a minimum weight maximal matching, where weights are on the vertices and the weight of a matching is defined as the sum of the weights of the saturated vertices. We call this problem minimum vertex-weight maximal matching, and abbreviate it as MVWMM.

MMM has been studied extensively in the literature. Most of the existing work concentrates on the complexity of MMM on special graph classes. While the problem of finding a maximum matching on a given graph is polynomially solvable by Edmonds's augmenting path algorithm [11], MMM is NP-hard on general graphs [15] and on several restricted graph classes. Examples include bipartite or planar graphs with maximum degree 3 [33], planar bipartite graphs, planar cubic graphs [17] and k-regular bipartite graphs for any fixed $k \geq 3$ [8]. In contrast, MMM is polynomially solvable in certain restricted graph classes. Examples include trees [24], block graphs [18], series-parallel graphs [26], bipartite permutation graphs and co-triangulated graphs [29]. Various approximation algorithms for MMM and MWMM have been proposed in the literature (see for instance [4, 9, 13, 14, 16, 23, 28]). Another line of research on MMM and MWMM considers the development of exponential time exact combinatorial algorithms [12, 32]. However, these algorithms have only been analyzed from a theoretical point of view, and no experimental study was conducted to evaluate their performance in practice.

The minimum edge dominating set problem (EDS) is closely related to MMM. We say that an edge in graph $G = (V, E)$ *dominates* itself and all edges sharing an end-vertex with it. EDS is defined as the problem of finding a minimum cardinality set of edges that dominates all edges in E. The relationship between MMM and EDS can be observed by noting that in a maximal matching of a graph $G = (V, E)$ each edge in E is necessarily dominated. In fact, it is known that the size of a minimum maximal matching is equal to the size of a minimum edge dominating set (hence, our algorithm for solving MMM can also be used to solve EDS optimally). Furthermore, an optimal solution of MMM can easily be generated from an optimal solution of EDS [33]. However, such a close relationship does not exist between weighted versions of the two problems [14].

To the best of our knowledge, MMM and MWMM have been addressed from mathematical programming point of view only in [31], where the authors formulate MWMM as an integer programming problem, derive

some valid inequalities and variable fixing rules, and test the efficacy of their formulation empirically. Integer programming formulations have also been used to derive approximation algorithms for weighted EDS [14] and its generalization, where edges have demands and capacities [3]. However, these methods cannot be used to solve MWMM due to the presence of edge weights [14].

In this paper, we start with an integer programming formulation of MWMM proposed in [31] and develop a decomposition algorithm based on Benders decomposition for its solution. Benders decomposition is widely used for solving large-scale mixed-integer programming problems (MIP). Instead of directly solving the MIP, Benders decomposition partitions it into a master problem that contains the integer variables, and a subproblem that contains the continuous variables. It then solves the master problem and the subproblem iteratively, adding cuts derived from linear programming duality theory to the master problem in each iteration. We refer the reader to [2] and [30] for details on Benders decomposition, and to [1, 6, 7, 25] for some applications of Benders decomposition within the context of optimization on graphs and networks. Our solution procedure decomposes the formulation [31] into a master problem, which seeks a vertex cover on the graph, and a subproblem, which seeks a perfect matching in the subgraph induced by the vertex cover selected by the master problem. While a straightforward formulation of our subproblem contains binary variables, we reformulate it as a linear programming problem having an exponential number of constraints. We then solve it via a combinatorial matching algorithm, and derive Benders cuts based on the combinatorial solution of the subproblem.

The rest of this paper is organized as follows. In Section 2 we give the basic integer programming (IP) formulation of MWMM used in [31] and point out some key observations, which relate maximal matchings to vertex covers in a graph. Section 3 explains our decomposition procedure developed to solve MMM and the generation of Benders feasibility cuts using Gallai-Edmonds decomposition. Section 4 is devoted to the development of our decomposition procedure for MWMM, where not only Benders feasibility cuts but also Benders optimality cuts are generated. We propose various approaches to improve the efficacy of our decomposition procedures for MMM and MWMM in Section 5. We present the results of our computational experiments in Section 6. Finally, we conclude our paper in Section 7 with a brief summary of our study and some future research directions.

2 Preliminaries

Let $G = (V, E)$ be a graph with vertex set V and edge set E. For a subset $V' \subseteq V$, we denote the subgraph induced by V' by $G[V']$. Similarly, let y be a $|V|$-dimensional binary vector. We denote the subgraph $G[\{i \in V | y_i = 1\}]$ by $G[y]$. For a vertex $v \in V$, let $N(v)$ denote the neighborhood of v, that is the set of vertices adjacent to v, and for $V' \subseteq V$ we have $N(V') = \cup_{v \in V'} N(v)$. An independent set I is a subset of V such that all vertices in I are pairwise non-adjacent, and a vertex cover S is a subset of V such that all edges have at least one end-vertex in S.

Let us first introduce the IP formulation used in [31] for MWMM. Let $G = (V, E)$ be an undirected graph and let c_{ij} denote the weight associated with edge $(i, j) \in E$, where edge weights c_{ij} can be positive, negative or zero. Taşkın and Ekim [31] define binary variable $x_{ij} = 1$ if edge $(i, j) \in E$ is selected in an optimal MWMM, and 0 otherwise. They also define binary variable $y_i = 1$ if vertex $i \in V$ is saturated by the matching (that is $\sum_{j \in N(i)} x_{ij} = 1$), and $y_i = 0$ if i is exposed (that is $\sum_{j \in N(i)} x_{ij} = 0$). Using these variables, they formulate MWMM as:

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

$$\text{subject to:} \quad \sum_{j \in N(i)} x_{ij} = y_i \quad \forall i \in V \tag{1b}$$

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

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

$$y_i \in \{0, 1\} \quad \forall i \in V. \tag{1e}$$

The objective function (1a) minimizes the total weight corresponding to the selected edges. Constraints (1b) enforce the condition that vertex i is saturated ($y_i = 1$) if some edge emanating from it is selected and it is exposed ($y_i = 0$) otherwise. Since y-variables are binary variables, (1b) also guarantees that the set of edges (i, j) such that $x_{ij} = 1$ forms a matching. Constraints (1c) enforce that if $x_{ij} = 1$ for some $(i, j) \in E$, then $y_i = y_j = 1$. Otherwise, if $x_{ij} = 0$ for some $(i, j) \in E$, then at least one of y_i and y_j still has to be saturated to ensure maximality of the matching. Note that since the x-variables are binary-valued, then (1b) will force y-variables to take on binary values. Therefore, y-variables can be relaxed as continuous. In [31], Model 1 is solved directly by adding some valid inequalities and applying a variable fixing rule. In this paper, we will focus on developing an efficient decomposition procedure for solving Model 1.

We first note that constraints (1c) reveal a vertex cover structure of the problem. Our critical observation is that for any maximal matching M of a given graph G, the set of vertices exposed with respect to M forms an independent set, and the set of vertices saturated by M forms a vertex cover of G. Therefore, MMM can be viewed as the problem of finding a vertex cover S of minimum cardinality such that $G[S]$ admits a perfect matching. Given a vertex cover S, let us denote the weight of a minimum weight perfect matching in $G[S]$ by $w(S)$, where $w(S) = \infty$ if $G[S]$ does not admit a perfect matching. Similar to MMM, MWMM can be seen as the problem of finding a vertex cover S such that $w(S)$ is minimized among all vertex covers of G. Based on these observations, we will first focus on MMM in Section 3, and postpone our treatment of the general case of MWMM until Section 4.

3 A Decomposition Approach for MMM

In this section we focus on deriving a decomposition algorithm for solving MMM, which is a special case of MWMM having $c_{ij} = 1$ for all edges. We first observe that Model 1 can be reformulated in terms of the y-variables for the case of MMM as follows:

$$\text{Model 2:} \quad \text{Minimize} \sum_{i \in V} y_i \tag{2a}$$

$$\text{subject to:} \; y_i + y_j \geq 1 \quad \forall (i,j) \; \in E \tag{2b}$$

$$G[y] \text{ admits a perfect matching} \tag{2c}$$

$$y_i \in \{0,1\} \quad \forall i \; \in V. \tag{2d}$$

Note that Model 1 contains $|V| + |E|$ binary variables, while Model 2 contains only $|V|$ binary variables. Even if the y-variables in Model 1 are relaxed as continuous, Model 1 still has $|E|$ binary variables. Since $|E| = O(|V|^2)$ for dense graphs, Model 2 contains significantly fewer binary variables than Model 1, which is advantageous from a computational point of view. Also note that Model 2 can be viewed as an integer programming formulation of the minimum vertex cover problem with some side constraints. These observations constitute the basis of our decomposition approach.

3.1 Solution Procedure

Model 2 is not well defined in its present form due to constraint (2c). In order to express (2c) as a set of linear inequalities, we first observe that given a binary $\hat{y}$-vector that represents a vertex cover such that (2b)

is satisfied, feasibility of (2c) can be checked by solving the following subproblem:

$$\text{SP}(\hat{y}): \quad \text{Minimize} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{3a}$$

$$\text{subject to:} \quad \sum_{j\in N(i)} x_{ij} = \hat{y}_i \quad \forall i \in V \tag{3b}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \ \in E, \tag{3c}$$

which is obtained from Model 1 for a fixed $\hat{y}$-vector. Note that the objective function of SP($\hat{y}$) does not matter since the role of SP($\hat{y}$) is just to check the existence of a perfect matching; if such a matching exists for some $\hat{y}$ then its cardinality is the same for all feasible solutions of SP($\hat{y}$). However, we prefer to use the minimization of the total weight of the selected edges (recall that $c_{ij} = 1$ for all edges in MMM) since this objective function will also be valid for MWMM. If SP($\hat{y}$) yields a feasible solution $\hat{x}$, then $\hat{y}$ is a feasible solution of Model 2, and represents the set of saturated vertices in a maximal matching. Furthermore, if $\hat{y}$ corresponds to a minimum vertex cover, then $\hat{y}$ solves Model 2 optimally and hence $\hat{x}$ represents a minimum maximal matching. On the other hand, if SP($\hat{y}$) is infeasible for some $\hat{y}$, then $G[\hat{y}]$ does not admit a perfect matching. In this case, the value of at least one y-variable has to be different in all feasible solutions of Model 2. Therefore, in principle, (2c) can be written as a collection of inequalities of form

$$\sum_{\{i\in V|\hat{y}_i=1\}} (1 - y_i) + \sum_{\{i\in V|\hat{y}_i=0\}} y_i \geq 1, \tag{4}$$

one for each $\hat{y}$ such that SP($\hat{y}$) has no feasible solution. Since there is an exponential number of constraints (4), it is not practical to enumerate them. Instead, they can be generated in a cutting-plane fashion as follows: we first relax (2c) and solve Model 2 to optimality. Let $\hat{y}$ denote an optimal solution. If SP($\hat{y}$) is infeasible, we then add the constraint (4) corresponding to $\hat{y}$ to Model 2, and re-solve it. Otherwise, if SP($\hat{y}$) yields a feasible solution $\hat{x}$, the set of edges $(i,j) \in E$ having $\hat{x}_{ij} = 1$ constitutes a minimum maximal matching, and we stop. Note that (4) corresponds to Laporte and Louveaux's feasibility cut for solving stochastic integer programs [20]. Also note that Model 2 is guaranteed to be feasible since any maximal matching on G yields a feasible solution of the problem. Furthermore, since a $\hat{y}$-vector is never considered more than once, the naive algorithm described above terminates in a finite number of iterations.

3.2 Benders Feasibility Cuts Using Gallai-Edmonds Decomposition

The naive algorithm discussed in the previous section can be improved in various ways. We first observe that SP($\hat{y}$) is an integer programming formulation, whose solution in general can take an exponential amount of time. However, recall that the role of SP($\hat{y}$) is only to check the existence of a perfect matching, which is a problem that can be solved in polynomial time [11]. In particular, given a $\hat{y}$-vector, we can use Edmonds's augmenting path algorithm [11] to seek a perfect matching in $G[\hat{y}]$. If $G[\hat{y}]$ does not admit a perfect matching, we can add a constraint (4) to Model 2, and re-solve it as before. However, note that each constraint of type (4) eliminates only a single solution from the feasible region, and hence constraints (4) are very weak. We next discuss how stronger cuts can be obtained.

At this stage, let us introduce the Gallai-Edmonds decomposition theorem (see [22] for more details). We first define a few key concepts that will be useful in our discussion. A graph is called *factor-critical* if it does not admit a perfect matching, but the removal of any single vertex leaves a graph that admits a perfect matching. A bipartite graph G with bipartition A and B is said to have positive surplus (as viewed from A) if the number of neighbors of X is larger than the size of X for any non-empty subset X of A.

Let $G = (V, E)$ be any graph. Let us denote the set of all vertices $v \in V$ such that there is a maximum matching that does not saturate v by $D(G)$. Let $A(G)$ be the set of vertices in $V \setminus D(G)$ that are adjacent to at least one vertex in $D(G)$. Finally, let $C(G) = V \setminus (A(G) \cup D(G))$ denote the set of all remaining vertices. The following theorem gives a very important characterization of a graph based on the structure of its maximum matchings.

Theorem 1. *(Gallai-Edmonds decomposition) [22] If G is a graph and $D(G), A(G)$ and $C(G)$ are defined as above, then:*

1. *the connected components of the subgraph induced by $D(G)$ are factor-critical,*
2. *the subgraph induced by $C(G)$ has a perfect matching,*
3. *the bipartite graph obtained from G by deleting the vertices of $C(G)$ and the edges induced by $A(G)$, and by contracting each connected component of $D(G)$ to a single vertex has positive surplus (as viewed from $A(G)$).*

The Gallai-Edmonds decomposition is unique for a given graph, and it can be calculated in polynomial time [22]. Note that if a graph G admits a perfect matching, then sets $A(G)$ and $D(G)$ are empty. An

example of the Gallai-Edmonds decomposition of a graph that does not admit a perfect matching is given in Figure 1.

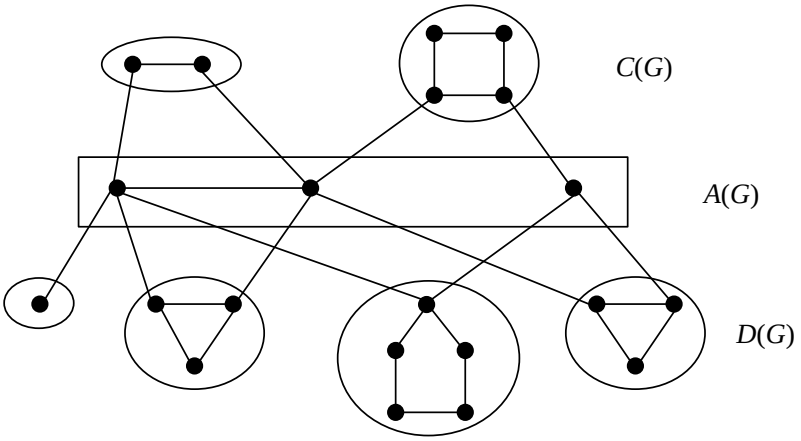

Figure 1: An example for Gallai-Edmonds decomposition.

Let $\hat{y}$ denote a vector such that $G[\hat{y}]$ does not admit a perfect matching. In order to obtain a constraint that can be used instead of (4) via the Gallai-Edmonds decomposition of $G[\hat{y}]$, let us first rewrite SP($\hat{y}$) as follows:

$$\text{SP1}(\hat{y})\text{: Minimize} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{5a}$$

$$\text{subject to:} \quad \sum_{j\in N(i)} x_{ij} = \hat{y}_i \quad \forall i \in V \tag{5b}$$

$$\sum_{ij\in o} x_{ij} \leq (|o|-1)/2 \quad \forall o \in OC \tag{5c}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in E, \tag{5d}$$

where OC is the set of all odd cardinality subsets of the vertices in G of size at least 3. It is well known that the x-variables can be relaxed as continuous due to the addition of constraints (5c) [11]. Note that SP1($\hat{y}$) is a linear program having an exponential number of constraints. Let π_i and θ_o denote the dual variables associated with constraints (5b) and (5c), respectively. By taking the dual of SP1($\hat{y}$), we obtain DSP1($\hat{y}$)

where $OC(ij)$ denotes the set of all odd cardinality subsets including both vertices i and j:

$$\text{DSP1}(\hat{y})\text{: Maximize } \sum_{i\in V}\pi_i\hat{y}_i + \sum_{o\in OC}((|o|-1)/2)\theta_o \tag{6a}$$

$$\text{subject to: } \pi_i + \pi_j + \sum_{o\in OC(ij)}\theta_o \leq c_{ij} \quad \forall (i,j)\in E \tag{6b}$$

$$\pi_i \text{ unrestricted} \quad \forall i \in V \tag{6c}$$

$$\theta_o \leq 0 \quad \forall o \in OC. \tag{6d}$$

We first note that DSP1($\hat{y}$) is feasible for any $\hat{y}$ since setting $\pi_i = \min_{j\in N(i)} c_{ij}/2$ for all $i \in V$ and $\theta_o = 0$ for all $o \in OC$ yields a feasible solution. Therefore, if SP1($\hat{y}$) is infeasible then DSP1($\hat{y}$) is necessarily unbounded. Let $\hat{y}$ represent a binary vector such that SP1($\hat{y}$) has no feasible solution and consequently the graph $G[\hat{y}]$ has no perfect matching. We consider the Gallai-Edmonds decomposition of $\hat{G} = G[\hat{y}]$. By definition, $D(\hat{G})$ is not empty. Let us denote the set of factor-critical components in $D(\hat{G})$ by $FCD(\hat{G})$. It follows from Theorem 1 item 3 that the cardinality of $FCD(\hat{G})$ is strictly greater than the cardinality of $A(\hat{G})$. Also, it follows from the definition of factor-criticality that every connected component of $D(\hat{G})$ has odd cardinality.

Proposition 1. *The following dual direction $(\bar{\pi}, \bar{\theta})$ is an unbounded ray for DSP1($\hat{y}$):*

$$\bar{\pi}_i = \begin{cases} 1 & \forall i \in D(\hat{G}) \\ -1 & \forall i \in N(D(\hat{G})) \setminus D(\hat{G}) \\ 0 & \textit{otherwise} \end{cases}$$

$$\bar{\theta}_o = \begin{cases} -2 & \forall o \in FCD(\hat{G}) \\ 0 & \textit{otherwise} \end{cases}$$

Proof. We need to show that adding $(\bar{\pi}, \bar{\theta})$ to any feasible solution with any positive coefficient yields a feasible solution with an improved objective function value for DSP1($\hat{y}$). To this end, we note that constraints (6c) and (6d) are trivially satisfied, and the term added by this direction to the left hand side of inequality (6b) is non-positive for all edges. In fact, the only case where positive terms are involved for edge (i, j) is when at least one of i and j is in $D(\hat{G})$. If both i and j are in $D(\hat{G})$, then (i, j) belongs to a connected component of $D(\hat{G})$ and hence $\bar{\pi}_i = 1, \bar{\pi}_j = 1$ and the corresponding odd cardinality subset containing edge (i, j) has $\bar{\theta}_o = -2$. If $i \in D(\hat{G})$ and $j \notin D(\hat{G})$ then $\bar{\pi}_i = 1, \bar{\pi}_j = -1$ and all odd cardinality subsets containing edge

(i,j) have $\bar{\theta}_o = 0$ since they are not entirely included in $D(\hat{G})$.

Furthermore, one can observe that the value of the objective function for a solution on this ray can be increased without bound since the term added to the objective function by this direction is strictly positive. In particular, the change in the objective function along the ray $(\bar{\pi}, \bar{\theta})$ is a positive multiple of:

$$\begin{aligned}
&\sum_{i \in D(\hat{G})} \hat{y}_i - \sum_{i \in N(D(\hat{G})) \setminus D(\hat{G})} \hat{y}_i - \sum_{o \in FCD(\hat{G})} (|o| - 1) \\
&= \sum_{i \in D(\hat{G})} 1 - \sum_{i \in A(\hat{G})} 1 - \sum_{o \in FCD(\hat{G})} (|o| - 1) \\
&= \left(|D(\hat{G})| - \sum_{o \in FCD(\hat{G})} (|o| - 1) \right) - |A(\hat{G})|
\end{aligned}$$

This added term is equal to a positive multiple of $(a - b)$, where a is equal to the number of connected components in $D(\hat{G})$ and b is the cardinality of $A(\hat{G})$. By Theorem 1 item 3 there are strictly more connected components in $D(\hat{G})$ than the cardinality of $A(\hat{G})$; therefore $(a - b)$ is strictly positive. □

Consider a $\hat{y}$-vector such that $G[\hat{y}]$ does not admit a perfect matching, and let $(\bar{\pi}, \bar{\theta})$ denote a dual direction for DSP1($\hat{y}$) as defined in Proposition 1. Since $(\bar{\pi}, \bar{\theta})$ is an unbounded direction for DSP1($\hat{y}$), $\sum_{i \in V} \bar{\pi}_i \hat{y}_i + \sum_{o \in OC} ((|o| - 1)/2) \bar{\theta}_o > 0$, and SP1($\hat{y}$) is infeasible. Similarly, SP1(y) is infeasible for all y such that $\sum_{i \in V} \bar{\pi}_i y_i + \sum_{o \in OC} ((|o| - 1)/2) \bar{\theta}_o > 0$. Therefore, the following inequality should be satisfied for feasibility of SP1(y), and hence can be used as the Benders feasibility cut

$$\sum_{i \in D(\hat{G})} y_i - \sum_{i \in N(D(\hat{G})) \setminus D(\hat{G})} y_i \leq \sum_{o \in FCD(\hat{G})} (|o| - 1), \tag{7}$$

where the right hand side is equal to the number of saturated vertices in $D(\hat{G})$ in a maximum matching of the subgraph $\hat{G}$. This cut implies that the number of saturated vertices in $N(D(\hat{G})) \setminus D(\hat{G})$ has to be greater than or equal to the number of connected components in $D(\hat{G})$ in order for all vertices in $D(\hat{G})$ to be saturated. This constraint is a necessary condition for saturating all vertices in $D(\hat{G})$ and clearly is not satisfied by the solution $\hat{y}$. Therefore, it can be used instead of (4). Indeed, as shown in what follows, (7) is stronger than (4) because while (4) eliminates a single y-vector from the solution space, (7) is based on a precise reason of infeasibility of SP1($\hat{y}$), and eliminates simultaneously many y-vectors.

Proposition 2. *Feasibility cuts* (7) *are stronger than feasibility cuts* (4).

Proof. Let us define two polyhedra as follows:

$$P_4 := \{y \in [0,1]^{|V|} \mid y_i + y_j \geq 1, \forall (i,j) \in E \text{ and } \sum_{\{i \in V | \hat{y}_i = 1\}} (1 - y_i) + \sum_{\{i \in V | \hat{y}_i = 0\}} y_i \geq 1,$$
$$\forall \hat{y} \in \{0,1\}^{|V|} \text{ s.t. } G[\hat{y}] \text{ has no perfect matching}\}$$

$$P_7 := \{y \in [0,1]^{|V|} \mid y_i + y_j \geq 1, \forall (i,j) \in E \text{ and } \sum_{i \in D(G[\hat{y}])} y_i - \sum_{i \in N(D(G[\hat{y}])) \setminus D(G[\hat{y}])} y_i \leq \sum_{o \in FCD(G[\hat{y}])} (|o| - 1),$$
$$\forall \hat{y} \in \{0,1\}^{|V|} \text{ s.t. } G[\hat{y}] \text{ has no perfect matching}\}$$

To show that $P_7 \subseteq P_4$, let us show that $\bar{y} \in P_4$ for any $\bar{y} \in P_7$. Since the vertex cover constraints are common on both polyhedra $\bar{y}_i + \bar{y}_j \geq 1, \forall (i,j) \in E$ holds automatically. Now, let $\hat{y} \in \{0,1\}^{|V|}$ s.t. $G[\hat{y}]$ has no perfect matching. For simplicity, denote $\hat{G} := G[\hat{y}]$ and $V_{\hat{G}} := \{i \in V | \hat{y}_i = 1\}$, and let $N(D(\hat{G})) \setminus D(\hat{G}) = A(\hat{G}) \cup \hat{X}$, where $\hat{X} := N(D(\hat{G})) \setminus (D(\hat{G}) \cup A(\hat{G}))$. We will show that $\sum_{i \in V_{\hat{G}}} (1 - \bar{y}_i) + \sum_{i \in V \setminus V_{\hat{G}}} \bar{y}_i \geq 1$, which can also be written as $\sum_{i \in V_{\hat{G}}} \bar{y}_i - \sum_{i \in V \setminus V_{\hat{G}}} \bar{y}_i \leq |V_{\hat{G}}| - 1$. To this end, since $\bar{y}_i \geq 0, \forall i \in V$ and $\hat{X} \subseteq (V \setminus V_{\hat{G}})$, it suffices to show that $\sum_{i \in V_{\hat{G}}} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq |V_{\hat{G}}| - 1$. Consider the unique Gallai-Edmonds decomposition of $\hat{G}$. Note that the sets $D(\hat{G}), A(\hat{G}), C(\hat{G}), \hat{X}$ are all disjoint. Since there is no perfect matching in $\hat{G}$ and $\bar{y} \in P_7$, we have

$$\sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in A(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq \sum_{o \in FCD(\hat{G})} (|o| - 1)$$
$$\implies \sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in A(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq \sum_{o \in FCD(\hat{G})} |o| - |FCD(\hat{G})| = |D(\hat{G})| - |FCD(\hat{G})|$$

As noted earlier, it follows from Theorem 1 item 3 that $|FCD(\hat{G})| > |A(\hat{G})|$, i.e., $|FCD(\hat{G})| \geq |A(\hat{G})| + 1$.

$$\implies \sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in A(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq |D(\hat{G})| - |A(\hat{G})| - 1$$
$$\implies \sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq |D(\hat{G})| - |A(\hat{G})| - 1 + \sum_{i \in A(\hat{G})} \bar{y}_i$$
$$\implies \sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq |D(\hat{G})| - |A(\hat{G})| - 1 + |A(\hat{G})| \quad \text{since } \bar{y}_i \leq 1, \forall i \in V$$
$$\implies \sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq |D(\hat{G})| - 1 \ (\star)$$

We also have $\sum_{i \in C(\hat{G})} \bar{y}_i \leq |C(\hat{G})|$ and $\sum_{i \in A(\hat{G})} \bar{y}_i \leq |A(\hat{G})|$ $(\star\star)$ because $\bar{y}_i \leq 1, \forall i \in V$. Then,

$$\sum_{i \in V_{\hat{G}}} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i = \sum_{i \in C(\hat{G})} \bar{y}_i + \sum_{i \in A(\hat{G})} \bar{y}_i + \sum_{i \in D(\hat{G})} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq \sum_{i \in C(\hat{G})} \bar{y}_i + \sum_{i \in A(\hat{G})} \bar{y}_i + |D(\hat{G})| - 1 \text{ from } (\star)$$
$$\implies \sum_{i \in V_{\hat{G}}} \bar{y}_i - \sum_{i \in \hat{X}} \bar{y}_i \leq |C(\hat{G})| + |A(\hat{G})| + |D(\hat{G})| - 1 = |V_{\hat{G}}| - 1 \text{ from } (\star\star)$$

Let us next show that this containment is strict, that is, there is at least one point in P_4 which is not in P_7. For this purpose, consider the graph $G = (V, E)$ where $V = \{1, 2, 3\}$ and $E = \{(1,3), (2,3)\}$, and the point $(0.5, 0.5, 0.5)$. This point is clearly in P_4. Consider the vertex cover $\hat{y}_1 = \hat{y}_2 = \hat{y}_3 = 1$, for which $\hat{G}$ admits no perfect matching. The related Gallai-Edmonds decomposition has two factor critical components in $D(\hat{G})$: $\{1\}$ and $\{2\}$. The corresponding constraint of type (7), $y_1 + y_2 - y_3 \leq 0$, belongs to P_7, and is

violated by the point $(0.5, 0.5, 0.5)$. Therefore $(0.5, 0.5, 0.5) \notin P_7$, and $P_7 \subset P_4$. □

Algorithm 1 summarizes our solution procedure for MMM:

Algorithm 1 MMM Benders Decomposition

Ensure: A minimum maximal matching
Require: A graph $G = (V, E)$
1: Solve Model 2 with (2c) relaxed. Let $\hat{y}$ be an optimal solution.
2: Find a maximum matching M in $G[\hat{y}]$ {Use Edmonds's Maximum Matching Algorithm}
3: **if** M is a perfect matching of $G[\hat{y}]$ **then**
4: M is a minimum maximal matching of G, STOP
5: **else**
6: Generate feasibility cut (7) for $\hat{y}$, add it to Model 2 and go to step 1
7: **end if**

Remark 1. Algorithm 1 can also be used for solving minimum vertex-weight maximal matching (MVWMM), which is a generalization of MMM such that there are weights on vertices (but not on edges). Let the weight of vertex $i \in V$ be given by w_i. Algorithm 1 can be used to solve MVWMM with the following slight modification to the objective function of Model 2:

$$\text{Minimize} \sum_{i \in V} w_i y_i. \tag{8}$$

4 A Decomposition Approach for MWMM

In this section, we extend our analysis to the minimum weight maximal matching (MWMM) problem. Recall that in MWMM each edge $(i, j) \in E$ has a weight c_{ij}, and the objective is to minimize the total edge weight in a maximal matching. MMM is a special case of MWMM where $c_{ij} = 1$ for all $(i, j) \in E$. Furthermore, an instance of MVWMM can be transformed into an instance of MWMM by setting $c_{ij} = w_i + w_j$ for all $(i, j) \in E$. After this transformation, an optimal solution of MWMM is also an optimal solution of MVWMM with the same objective function value. Therefore, MWMM is a generalization of both MMM and MVWMM.

Similar to our analysis of MMM, we first observe that Model 1 can be reformulated for the case of

MWMM as:

$$\text{Model 3:} \quad \text{Minimize } t \tag{9a}$$

$$\text{subject to: } y_i + y_j \geq 1 \quad \forall (i,j) \ \in E \tag{9b}$$

$$G[y] \text{ admits a perfect matching with total weight } t \tag{9c}$$

$$y_i \in \{0,1\} \quad \forall i \ \in V \tag{9d}$$

$$t \geq LB, \tag{9e}$$

where LB is a lower bound on the weight of any maximal matching. A valid value for LB can be calculated in polynomial time by finding a minimum weight matching on G; this can be done simply by multiplying all weights by -1 and finding a maximum weight matching. Note that if all edge weights c_{ij} are non-negative, then the empty matching is a minimum weight matching, and hence $LB = 0$.

Similar to our solution procedure for MMM, we first need to express (9c) as a set of linear inequalities. Given a $\hat{y}$-vector that represents a vertex cover and a $\hat{t}$ value, the satisfaction of (9c) can be checked implicitly by seeking a minimum weight perfect matching on $G[\hat{y}]$, which can be calculated in polynomial time [22]. If $G[\hat{y}]$ does not admit a perfect matching, then its Gallai-Edmonds decomposition can be calculated and a Benders feasibility cut (7) can be generated as before. Otherwise, let $t^*[\hat{y}]$ denote the weight of a minimum weight perfect matching in $G[\hat{y}]$. If $\hat{t} \geq t^*[\hat{y}]$, then $(\hat{y}, \hat{t})$ is a feasible solution of Model 3. Otherwise, the following constraint is valid:

$$t \geq t^*[\hat{y}] - (t^*[\hat{y}] - LB) \left(\sum_{\{i \in V | \hat{y}_i = 1\}} (1 - y_i) + \sum_{\{i \in V | \hat{y}_i = 0\}} y_i \right). \tag{10}$$

Note that (10) reduces to $t \geq t^*[\hat{y}]$ for $y = \hat{y}$, and is redundant for all $y \neq \hat{y}$. In principle, it is possible to express (9c) as a combination of (4) or (7) (one for each $\hat{y}$ that does not admit a perfect matching) and (10) (one for each $\hat{y}$ that admits a perfect matching), and generate them as needed. However, since each constraint (10) is redundant for all y except the $\hat{y}$-vector that it is generated for, it may be required to enumerate all $\hat{y}$ that yield a perfect matching before solving the problem to optimality. Furthermore, we observe that $(t = LB, y_i = 0.5, \ \forall i \in V)$ is a fractional solution that satisfies all (10) constraints. Hence, the linear programming lower bound is not improved by the addition of (10). Note that (10) corresponds to Laporte and Louveaux's optimality cut for solving stochastic integer programs [20]. Similar observations about the weakness of (10) have been made by various authors within the context of stochastic integer

programming (see [19, 21, 27]). We will next discuss how to extract dual information from the subproblem to generate cuts that can be used instead of (10).

Let us first write the following equivalent formulation for SP1($\hat{y}$), where we simply reduce the problem of finding a minimum weight perfect matching to the subgraph $G[\hat{y}] = (V[\hat{y}], E[\hat{y}])$. We denote by $OC[\hat{y}]$ the set of all odd cardinality subsets whose vertices are all included in $G[\hat{y}]$.

$$\text{SP2}(\hat{y})\text{: Minimize} \quad \sum_{(i,j)\in \hat{E}[\hat{y}]} c_{ij}x_{ij} \tag{11a}$$

$$\text{subject to:} \quad \sum_{j\in N_{G[\hat{y}]}(i)} x_{ij} = 1 \quad \forall i \in V[\hat{y}] \tag{11b}$$

$$\sum_{ij\in o} x_{ij} \leq (|o|-1)/2 \quad \forall o \in OC[\hat{y}] \tag{11c}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in E[\hat{y}]. \tag{11d}$$

Now, let π_i and θ_o denote the dual variables associated with (11b) and (11c), respectively. By taking the dual of SP2($\hat{y}$), we obtain DSP2($\hat{y}$):

$$\text{DSP2}(\hat{y})\text{: Maximize} \quad \sum_{i\in V[\hat{y}]} \pi_i + \sum_{o\in OC[\hat{y}]} ((|o|-1)/2)\theta_o \tag{12a}$$

$$\text{subject to:} \quad \pi_i + \pi_j + \sum_{o\in OC[\hat{y}](ij)} \theta_o \leq c_{ij} \quad \forall (i,j) \in E[\hat{y}] \tag{12b}$$

$$\pi_i \text{ unrestricted} \quad \forall i \in V[\hat{y}] \tag{12c}$$

$$\theta_o \leq 0 \quad \forall o \in OC[\hat{y}]. \tag{12d}$$

Let (π^*, θ^*) be an optimal solution of DSP2($\hat{y}$). In principle, such a dual optimal solution can be calculated by solving SP2($\hat{y}$) or DSP2($\hat{y}$) as linear programming problems. However, an optimal dual solution can also be calculated by the weighted version of Edmonds's algorithm [22].

Proposition 3. *Let (π^*, θ^*) be an optimal solution of DSP2($\hat{y}$). Then, the following solution (π^{**}, θ^{**}) is optimal for DSP1($\hat{y}$):*

$$\theta_o^{**} = \begin{cases} \theta_o^* & \textit{for} \quad o \in OC[\hat{y}] \\ 0 & \textit{for} \quad o \in OC \setminus OC[\hat{y}] \end{cases}$$

$$\pi_i^{**} = \begin{cases} \pi_i^* & for \quad i \in V[\hat{y}] \\ \min\limits_{(i,j)\in E} \{c_{ij} - \pi_j^*\} & for \quad i \in V \setminus V[\hat{y}] \end{cases}$$

Proof. First we show that (π^{**}, θ^{**}) is a feasible solution for DSP1($\hat{y}$). Constraints (6c) and (6d) are trivially satisfied due to the feasibility of (θ_o^*, π_i^*) for DSP2($\hat{y}$) and by definition of θ_o^{**} and π_i^{**} . So, it is enough to show that $(\theta_o^{**}, \pi_i^{**})$ satisfies constraints (6b). For each edge $(i,j) \in E$, let us examine the following cases:
Case 1: $i \notin V[\hat{y}]$, $j \notin V[\hat{y}]$. This case is not possible in any feasible solution of Model 3 because of the vertex cover constraint (9b).
Case 2: $i \in V[\hat{y}]$, $j \in V[\hat{y}]$. In this case we have $\pi_i^{**} = \pi_i^*$ and $\pi_j^{**} = \pi_j^*$ by definition. Also, $\theta_o^{**} = \theta_o^*$ for each $o \in OC(ij)$. Since (π^*, θ^*) satisfies (12b), (6b) is also satisfied.
Case 3: (WLOG) $i \notin V[\hat{y}]$, $j \in V[\hat{y}]$. In this case, for each odd cardinality subset $o \in OC$ containing edge (i,j), we have $\theta_o = 0$ since o is not included in $OC[\hat{y}]$. In addition, we have $\pi_j^{**} = \pi_j^*$ since $j \in V[\hat{y}]$, and we have $\pi_i^{**} \leq c_{ij} - \pi_j^*$ since $i \notin V[\hat{y}]$. Therefore, $\pi_i^{**} + \pi_j^{**} \leq c_{ij}$ and hence (6b) holds for edge (i,j).

Now, let us show that (π^{**}, θ^{**}) is an optimal solution of DSP1($\hat{y}$). First of all, note that the objective function values of DSP1($\hat{y}$) and DSP2($\hat{y}$) are the same. We also know by strong duality that the objective function values of SP2($\hat{y}$) and DSP2($\hat{y}$) are equal at optimality. Since SP1($\hat{y}$) and SP2($\hat{y}$) are equivalent, we get the conclusion that the objective values of SP1($\hat{y}$) and DSP1($\hat{y}$) are the same, so we have an optimal solution of DSP1($\hat{y}$). □

Consider a $\hat{y}$-vector such that $G[\hat{y}]$ admits a perfect matching, and let (π^{**}, θ^{**}) denote an optimal dual solution of DSP1($\hat{y}$) as defined in Proposition 3. Since the feasible region of DSP1(y) does not depend on y, (π^{**}, θ^{**}) is a feasible dual solution for any value of y, and hence $\sum_{i\in V} \pi_i^{**} y_i + \sum_{o\in OC}((|o|-1)/2)\theta_o^{**}$ yields a lower bound on the optimal objective function value of DSP1(y). Therefore, the following Benders optimality cut is valid:

$$t \geq \sum_{i\in V} \pi_i^{**} y_i + \sum_{o\in OC} ((|o|-1)/2)\theta_o^{**}. \tag{13}$$

Note that (13) provides a lower bound on t, which is clearly violated by the current $(\hat{y}, \hat{t})$ if $\hat{t} < t^*[\hat{y}] = \sum_{i\in V} \pi_i^{**} \hat{y}_i + \sum_{o\in OC}((|o|-1)/2)\theta_o^{**}$. Although a direct comparison of cuts (10) and (13) is not as straightforward as for the cuts (4) and (7), we will show in Section 6 by means of computational experiments that using optimality cuts (13) instead of optimality cuts (10) is much more efficient in practice.

Whenever $G[\hat{y}]$ admits a minimum weight perfect matching, it provides an upper bound UB on t, which

is updated if improved. Then, the corresponding optimality cut (13) is generated and added to Model 3, which is then re-solved. This procedure is repeated until the objective value of Model 3, which is a lower bound LB for t, is equal to the weight of a minimum weight perfect matching, which is an upper bound UB for t. The above solution procedure for MWMM can be summarized as in Algorithm 2.

Algorithm 2 MWMM Benders Decomposition

Require: A graph $G = (V, E)$ with edge weights c_{ij}
Ensure: A minimum weight maximal matching
1: Set $UB = \infty$
2: Solve Model 3 with (9c) relaxed. Let $(\hat{y}, \hat{t})$ be an optimal solution. Set LB=objective function value of Model 3
3: Seek a minimum weight perfect matching M in $G[\hat{y}]$
4: **if** M exists **then**
5: Set $UB = \min$ (UB, total weight of M)
6: **if** $LB = UB$ **then**
7: M is a minimum weight maximal matching, STOP
8: **end if**
9: Let (π^*, θ^*) denote an optimal dual solution associated with M
10: Compute an optimal solution (π^{**}, θ^{**}) of DSP1 as described in Proposition 3
11: Add optimality cut (13) generated for (π^{**}, θ^{**}) to Model 3, go to step 2
12: **else**
13: Generate feasibility cut (7) for $\hat{y}$, add it to Model 3 and go to step 2
14: **end if**

5 Modelling and Algorithmic Improvements

In our preliminary computational tests, we observed that repeatedly re-solving Models 2 and 3 to optimality in each iteration constitutes the bottleneck of our decomposition approach. This is not surprising since Models 2 and 3 are integer programming problems, whose solutions require an exponential amount of time. On the other hand, given a solution $\hat{y}$, calculation of a minimum weight perfect matching or identification of the Gallai-Edmonds decomposition can be performed in polynomial time [22]. In this section we discuss some approaches for improving the solvability of Models 2 and 3.

5.1 Initial solution

It is well known that the existence of a good initial feasible solution can help to improve the performance of integer programming solvers because it provides a good upper bound, which allows the solver to prune more branch-and-bound nodes and allows the solver to apply strategies such as reduced cost fixing. Providing such an initial solution can have a significant impact on solution performance [5]. To this end, we use the

greedy MMM and MWMM algorithms described in [31], which are based on choosing at each step an edge $(i,j) \in E$ with the lowest ratio of c_{ij} to the total weight of the edges (including itself) that cannot be in the matching once edge (i,j) is included in it.

5.2 Valid Inequalities

Another improvement in both Algorithms 1 and 2 can be obtained as follows. Recall that given a $\hat{y}$-vector that does not admit a perfect matching in $G[\hat{y}]$, a Benders feasibility cut (7) can be generated. Now, suppose that $G[\hat{y}]$ is disconnected (note that $G[\hat{y}]$ can be disconnected even though G is connected). In this case, $G[\hat{y}]$ admits a perfect matching if and only if each connected component of $G[\hat{y}]$ admits a perfect matching. Based on this observation, we first identify connected components of $G[\hat{y}]$, seek a perfect matching in each connected component separately, and generate an individual feasibility cut (7) for each connected component that does not admit a perfect matching.

One can also improve the formulations of Models 2 and 3 by deriving some valid inequalities. These valid inequalities state some necessary conditions that not only improve the lower bound of Models 2 and 3, but also allow our decomposition approach to converge in a fewer number of iterations by eliminating $\hat{y}$-vectors that cannot admit a perfect matching in $G[\hat{y}]$. Our first valid inequality is based on a very simple observation: an even number of vertices must be saturated in order for $G[\hat{y}]$ to admit a perfect matching. Hence, constraint (14) is valid:

$$\sum_{i \in V} y_i = 2k, \text{ where } k \text{ is an integer variable.} \tag{14}$$

A different set of valid inequalities can be derived as follows: We observe that if a vertex i is saturated, then at least one of its neighbors should also be saturated (by definition of a matching); and if a vertex i is not saturated, then all of its neighbors should be saturated in order to ensure maximality of the matching. Constraints (15), which are valid for both Model 2 and Model 3, are based on this observation.

$$\sum_{j \in N(i)} y_j + |N(i) - 1|\, y_i \geq |N(i)|\,, \quad \forall i \in V. \tag{15}$$

We observe that (15) generalizes the variable fixing rule proposed in [31]. In particular, let $i \in V$ be a vertex having degree 1, and let $j \in V$ be its only neighbor. Taşkın and Ekim [31] observe that since at least one of the vertices i and j has to be saturated in a maximal matching, and since the saturation of vertex i

implies that vertex j is also saturated by the matching, it follows that vertex j has to be saturated in any maximal matching. Therefore, they propose the following variable fixing rule:

$$y_j = 1 \quad \forall (i,j) \in E, |N(i)| = 1. \tag{16}$$

Note that (15) simplifies to $y_j \geq 1$ for each vertex i having $N(i) = \{j\}$, which is equivalent to (16) since y-variables are binary.

5.3 Single branch-and-bound-tree

We note that instead of solving the integer programming problems to optimality in each iteration, we can interrupt the branch-and-bound solution process each time the solver finds an integer solution $\hat{y}$ (and $\hat{t}$), and check whether a feasibility cut (7) (or optimality cut (13)) that is violated by the current integer solution can be generated. If no cuts can be generated, we accept the current solution as the new incumbent and resume the solution process. If some cuts are generated, we reject the current solution, add the newly generated cuts to the formulation, and again resume the solution process. In our tests, this approach consistently performed better than solving Models 2 and 3 to optimality in each iteration. This can be explained by noting that with this approach the problem is solved using a single branch-and-bound tree as opposed to repeatedly generating a branch-and-bound tree in each iteration.

5.4 Using MVWMM in the solution procedure for MWMM

Consider the initial iteration of Algorithm 2, where no cuts (7) or (13) have been generated. Initially any vertex cover provides an optimal solution of Model 3. Furthermore, the initial optimal value of t equals its lower bound LB, which provides a weak lower bound on the optimal objective function value. Therefore, it may take a long time for Algorithm 2 to converge to an optimal solution. In this section, we will focus on improving the convergence of Algorithm 2 by guiding it to promising vertex covers and improving its lower bound.

Recall that any instance of MVWMM can be converted into an instance of MWMM by choosing the edge weights appropriately (Section 4). In this section, we further investigate the relationship between the two problems. Let G_c be an instance of MWMM having edge weights c_{ij}. Assume that there is a set of vertex weights w_i (of unrestricted sign and possibly fractional) for each vertex i such that for each edge $(i,j) \in E$ we have $c_{ij} = w_i + w_j$. Then one can solve MWMM on G_c by considering the graph G_w where the weights

on vertices satisfy the above condition and by solving MVWMM on G_w as described in Remark 1. Note that in this situation the t-variable and optimality cuts (13) are no longer needed since any perfect matching in subgraph $G[\hat{y}]$ has the same total weight (equal to the sum of the weights of saturated vertices), which is already minimized by the objective function (8). On the other hand, if no set of vertex weights satisfies the above condition, then one cannot transform MWMM on G_c to an equivalent MVWMM instance. In this situation, however, we can distribute the edge weights to vertices "as much as possible" and use this information in the objective function of Model 3. Specifically, given an MWMM instance, we solve the following linear program, which tries to find weights w_i for vertices $i \in V$ such that for each edge $(i,j) \in E$ we have $w_i + w_j = c_{ij}$.

$$\text{VW: Minimize} \sum_{(i,j)\in E} s_{ij} \tag{17a}$$

$$\text{subject to: } w_i + w_j + s_{ij} = c_{ij} \quad \forall (i,j) \in E \tag{17b}$$

$$s_{ij} \geq 0 \quad \forall (i,j) \in E \tag{17c}$$

$$w_i \text{ unrestricted } \forall i \in V. \tag{17d}$$

If an optimal solution (w^*, s^*) of VW has objective function value zero, then the related MWMM instance can be seen as an MVWMM instance with vertex weights w^*, and hence can be solved using the procedure described in Remark 1. Otherwise, there is no set of vertex weights satisfying the equality $w_i + w_j = c_{ij}$ for all edges $(i,j) \in E$. Instead, VW returns a solution (w^*, s^*) such that $w_i^* + w_j^* \leq c_{ij}$ for all edges $(i,j) \in E$ and s_{ij}^* can be seen as the residual weight of the edge (i,j). This information can be incorporated by considering the graph G_{s^*} and replacing the objective function of Model 3 as follows:

$$\text{Minimize} \sum_{i\in V} w_i^* y_i + \bar{t}, \tag{18}$$

where $\bar{t}$ is a non-negative variable representing the contribution of the residual edge weights s_{ij}^* to the weight of the perfect matching that we seek in Model 3. With this new objective function, Model 3 chooses vertices to be saturated (vertices i having $\hat{y}_i = 1$) in such a way that their total weight is minimized. If the set of saturated vertices in G_{s^*} admits a perfect matching having total weight $\bar{t}$, then this matching provides a maximal matching in G whose total weight is equal to $\sum_{i\in V} w_i \hat{y}_i + \bar{t}$.

With this new objective function of Model 3 and the modified version of Algorithm 2, where the weights are first distributed to the vertices in the best possible way, the initial lower bound of Model 3 is improved

and hence the algorithm is likely to perform fewer iterations before proving optimality. This fact is also confirmed with our experiments (see Section 6). Moreover w^* is used to drive the objective function of Model 3 to find near-optimal vertex covers.

6 Computational Results

We first conduct a series of experiments to evaluate the efficiency of algorithmic improvements suggested in Section 5 as well as to evaluate the strength of the optimality cuts (13) as compared to the optimality cuts (10). These results will guide us to obtain a modified version of Algorithm 1, called MMM_Benders_Improved, and Algorithm 2, called MWMM_Benders_Improved. We will then compare the performance of our improved algorithms with Algorithms 1 and 2, and with the results obtained by the algorithms presented in [31].

We implemented all algorithms using CPLEX 12.2 for solving the linear and integer programming problems, and LEMON Graph Library 1.2.1 [10] for finding connected components, seeking minimum weight perfect matchings and calculating Gallai-Edmonds decompositions. We executed all algorithms (including the ones proposed in [31]) on a computer with a 2.27 GHz Intel Xeon CPU and 12 GB RAM. Our base test data set consists of randomly generated problem instances for which the expected edge density of the graph (measured as $D = \frac{2|E|}{|V|\times(|V|-1)}$) 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 vertices. Data sets used in our tests are available online at `http://www.ie.boun.edu.tr/~taskin/research.php`.

In Table 1 and Table 2, we measure the effect of the suggested improvements on Algorithm 1 and Algorithm 2, respectively. We use instances with average density $D = 0.3, 0.5$, and 0.7 and number of vertices $|V| = 150, \ldots, 190$. In each column, the following results on ten instances are presented: "Cuts:" the total number of feasibility and optimality (for MWMM) cuts generated for ten instances, "Gap:" the average final percentage optimality gap for instances that could not be solved within the allowed time limit of 1200 seconds (calculated as $(UB - LB)/UB$ where UB denotes the upper bound and LB denotes the lower bound), "Init Gap:" the average percentage optimality gap of solutions found by the greedy algorithm described in Section 5.1, "Time:" the average amount of time in seconds spent by each algorithm on the instances that were solved to optimality within the allowed time limit.

In Table 1, we report the following results; "MMM_Benders:" Algorithm 1, "Initial Heuristic:" Algorithm

Table 1: Effect of proposed improvements on solution performance for solving MMM

		MMM_Benders		Initial Heuristic			Valid Inequalities	
D	$\|V\|$	Cuts	Time	Init Gap	Cuts	Time	Cuts	Time
0.3	150	14	66.0	6.24	10	81.8	0	35.2
	160	15	121.4	5.32	12	147.5	0	69.7
	170	17	258.2	5.50	6	296.1	0	124.3
	180	10	458.6	5.29	15	521.8	2	207.9
	190	16	945.6	5.64	12	896.7	0	385.6
0.5	150	9	44.8	3.36	10	50.8	0	19.7
	160	11	68.3	3.80	5	79.6	2	27.4
	170	11	131.5	3.80	8	150.7	0	48.2
	180	12	227.3	3.35	8	266.8	0	88.0
	190	10	315.6	3.92	14	383.0	0	98.1
0.7	150	20	38.7	3.97	11	44.7	0	13.0
	160	12	54.6	2.84	8	64.0	0	16.1
	170	17	76.8	3.13	8	93.1	0	22.8
	180	18	110.3	2.74	5	137.9	0	36.2
	190	8	187.4	2.91	3	195.2	0	50.4

Table 2: Effect of proposed improvements on solution performance for solving MWMM

		MWMM_Benders		Optimality Cut (10)			Initial Heuristic			MVWMM		Valid Inequalities	
D	$\|V\|$	Cuts	Time	Cuts	Gap	Time	Init Gap	Cuts	Time	Cuts	Time	Cuts	Time
0.3	150	159	37.5	67466	100	-	28.06	127	40.5	155	26.06	118	18.12
	160	95	74.9	66605	100	-	29.62	96	91.9	109	56.64	78	45.05
	170	104	122.9	75725	100	-	28.84	103	127.8	121	102.99	68	72.56
	180	65	335.6	70471	100	-	25.95	75	347.1	105	239.74	51	163.50
	190	64	592.4	72763	100	-	25.72	97	659.9	82	507.63	51	324.78
0.5	150	29	43.3	64755	100	-	21.35	36	48.1	25	33.26	28	18.25
	160	22	65.6	61483	100	-	18.19	37	73.9	25	59.57	25	29.35
	170	24	107.4	59596	100	-	18.51	32	115.9	32	86.43	27	44.50
	180	35	209.7	61932	100	-	16.83	36	253.0	20	201.84	22	78.53
	190	25	329.9	58436	100	-	15.61	31	355.8	16	335.89	21	110.45
0.7	150	31	40.1	59407	100	-	13.12	30	45.2	19	35.79	22	13.33
	160	22	54.6	59217	100	-	15.85	35	65.5	17	50.37	20	16.35
	170	31	77.8	56177	100	-	15.22	39	95.2	25	73.25	20	22.94
	180	28	108.2	56438	100	-	14.52	36	128.9	19	92.30	20	35.88
	190	18	188.4	50623	100	-	12.44	32	190.5	13	154.34	20	50.49

1 fed with the initial solution obtained by the greedy algorithm described in Section 5.1, "Valid Inequalities:" Algorithm 1 augmented with valid inequalities (14) and (15) introduced in Section 5.2.

We observe from Table 1 that although the greedy algorithm is able to find solutions within an average 4.1% of optimality, total running time is increased by 9.8% because of the time spent by the heuristic algorithm at the beginning. In our tests we observed that CPLEX is able to quickly find very good feasible solutions even if one is not provided initially. It follows that the use of initial heuristic is not justified by our experiments. On the contrary, the valid inequalities (14) and (15) substantially decreases both the total number of cuts and the total running time (respectively by 98% and 59.9%). As a result, our improved algorithm MMM_Benders_Improved, summarized in Algorithm 3, contains the valid inequalities (14) and

(15) but not the initial heuristic.

Algorithm 3 MMM_Benders_Improved

Require: A graph $G = (V, E)$
Ensure: A minimum maximal matching
1: Run Algorithm 1 with cuts (14) and (15) added to Model 2 and feasibility cuts (7) added for each connected component of $G[\hat{y}]$.

Table 2 shows the following results; "MWMM_Benders:" Algorithm 2,"Optimality Cut (10):" Algorithm 2 where optimality cuts (13) in line 11 are replaced with optimality cuts (10), "Initial Heuristic": Algorithm 2 fed with the initial solution obtained by the greedy algorithm described in Section 5.1,"MVWMM:" Algorithm 2 where the edge weights are distributed to the vertices using the procedure described in Section 5.4, "Valid Inequalities:" Algorithm 2 augmented with valid inequalities (14) and (15) introduced in Section 5.2.

Our first observation is that, when we use cuts (10) instead of cuts (13), none of the instances can be solved in the given time limit. Therefore, the improved version of our algorithm contains our proposed optimality cuts (13), which utilize duality information from the solution of the subproblem. The use of the greedy algorithm to find an initial solution is once again not justified since it increases both the total number of cuts (by 11.9%) and the total running time (by 10.5%). We also observe that the initial heuristic gaps are higher than the ones for the unweighted case, in alignment with the results obtained in [31]. As for the contribution of the weight distribution to the vertices, although the total number of cuts generated slightly increases (by 4.1%), the total running time is improved by 13.9%. Finally, valid inequalities provide the most significant improvement as in the unweighted case: 21.4% reduction in the total number of cuts, 56.3% reduction in the total running time. Following these results, we implemented the modified version of Algorithm 2, called MWMM_Benders_Improved, which contains all the improvements described in Section 5 except the use of a greedy algorithm to obtain an initial solution. This algorithm is summarized in Algorithm 4.

Now, we compare the performance of our algorithms with the direct solution of the integer programming formulation proposed in [31], which is enhanced by valid inequalities and variable fixing rules. For this experiment, we executed our algorithms on the same problem instances as in [31] with average edge density $D = 0.3, 0.5,$ and 0.7 and number of vertices $|V| = 150, \ldots, 190$. In Table 3, the set of columns titled "Taşkın and Ekim (2012)" correspond to the results obtained by the algorithm presented in [31] and the set of columns titled "MMM_Benders" and "MMM_Benders_Improved" correspond to Algorithms 1 and 3, respectively. Note that all tests were performed on the same hardware and software environment (operating system, compiler and version of CPLEX). Furthermore, both codes were executed when no other tasks were

Algorithm 4 MWMM_Benders_Improved

Require: A graph $G = (V, E)$ with edge weights c_{ij}
Ensure: A minimum weight maximal matching
1: Solve VW. Let (w^*, s^*) be an optimal solution. Let G_w be a copy of G where vertex i has weight w_i^* and edge (i, j) has weight s_{ij}^*
2: **if** the optimal value of VW is zero **then**
3: Run Algorithm 3 for G_w where Model 2 has objective function $\min \sum_{i \in V} w_i y_i$ and return its solution.
4: **else**
5: Run Algorithm 2 for G_w where Model 3 has objective function $\min \sum_{i \in V} w_i y_i + \bar{t}$, cuts (14) and (15) are added and feasibility cuts (7) are added for each connected component of $G_w[\hat{y}]$. Return its solution.
6: **end if**

Table 3: Comparison of algorithms for solving MMM on medium-sized graphs

		Taşkın and Ekim (2012)			MMM_Benders			MMM_Benders_Improved		
D	$\|V\|$	Solved	Gap	Time	Solved	Gap	Time	Solved	Gap	Time
0.3	150	10	-	129.6	10	-	66.0	10	-	35.2
	160	10	-	231.9	10	-	121.4	10	-	69.7
	170	10	-	421.2	10	-	258.2	10	-	124.3
	180	10	-	712.0	10	-	458.6	10	-	207.9
	190	3	2.33	818.0	10	-	945.6	10	-	385.6
0.5	150	10	-	154.5	10	-	44.8	10	-	19.7
	160	10	-	215.5	10	-	68.3	10	-	27.4
	170	10	-	573.5	10	-	131.5	10	-	48.2
	180	9	1.19	857.4	10	-	227.3	10	-	88.0
	190	4	2.44	1018.6	10	-	315.6	10	-	98.1
0.7	150	10	-	320.7	10	-	38.7	10	-	13.0
	160	10	-	509.5	10	-	54.6	10	-	16.1
	170	9	2.47	728.4	10	-	76.8	10	-	22.8
	180	6	1.47	1015.8	10	-	110.3	10	-	36.2
	190	0	2.89	-	10	-	187.4	10	-	50.4

running on the computer so that results are comparable. 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) "Gap" and (iii) "Time" defined as previously.

We observe that while MMM_Benders and MMM_Benders_Improved can solve all 150 problem instances in this data set to optimality within a few minutes, CPLEX can only solve 121 instances to optimality within the enforced time limit. Comparing the problem instances that all approaches are able to solve to optimality, we observe that our decomposition algorithms significantly decrease the solution time in all instances. For instance, note that the integer programming approach can solve $D = 0.7, |V| = 160$ instances to optimality within an average of 509.5 seconds, while our improved algorithm is able to solve the same instances within an average of 16.1 seconds. We observe that while the performance of the integer programming formulation

given in [31] deteriorates rapidly as the number of vertices increases, the effect of $|V|$ is mitigated in our algorithms. We also observe that while the integer programming approach gets more difficult to solve as the graph density increases, perhaps surprisingly, the solution times for our decomposition approach decrease as D increases. This can be explained as follows: the integer programming formulation of [31] contains a binary variable and a constraint for each edge, and hence the size of the formulation increases significantly as D increases for a given $|V|$. However, our decomposition approach only contains a constraint for each edge in the master problem. Therefore, the number of variables in our approach is independent of the graph's edge density, and the number of constraints increases as D increases for a given $|V|$, which yields a tighter formulation that can be solved in a shorter amount of time. We also note that the improvements suggested in Section 5 significantly improve solution times on all tested instances.

Our second experiment compares our decomposition approach with the integer programming approach of [31] on graphs having weighted edges. Table 4 summarizes the results of this experiment. As before, we used the weighted problem instances used in [31] for this experiment. We observe that our decomposition approach significantly outperforms direct solution of the integer programming formulation in the weighted case, too. CPLEX is able to solve 133 instances to optimality within the allowed time limit while our algorithms can solve all 150 problem instances within a few minutes. Similar to the unweighted case, the performance of MWMM_Benders and MWMM_Benders_Improved deteriorates as $|V|$ increases, and improves as D increases. Comparing our two decomposition algorithms, we note again that our suggested improvements have a significant effect on solution times.

Comparing Tables 3 and 4, we observe that the integer programming formulation performs significantly better for the weighted problem instances. This is explained in [31] by noting that the existence of weights differentiates edges, hence decreasing the amount of symmetry in the model. On the other hand, the difference between solution times of unweighted and weighted instances is not very significant for our decomposition algorithms (with the notable exception of $D = 0.3$, which can be explained by noting that the effect of symmetry reduction due to weights becomes more apparent as D decreases). Recall that our master problem does not contain any variables for edges, and we account for edge weights implicitly via the solution of our subproblem. However, since we "transfer" as much weight from edges to vertices as possible (see Section 5.4), our approach also benefits indirectly from the existence of edge weights.

Our last experiment is aimed at analyzing the performance of our decomposition approach on larger graphs. For this experiment, we randomly generated larger graphs with up to $|V| = 300$ vertices. As before, we generated ten random problem instances for $D = 0.3, 0.5$ and 0.7, both with and without edge weights.

Table 4: Comparison of algorithms for solving MWMM on medium-sized graphs

		Taşkın and Ekim (2012)			MWMM_Benders			MWMM_Benders_Improved		
D	$\|V\|$	Solved	Gap	Time	Solved	Gap	Time	Solved	Gap	Time
0.3	150	10	-	43.4	10	-	37.5	10	-	15.1
	160	10	-	107.7	10	-	74.9	10	-	24.9
	170	10	-	123.0	10	-	122.9	10	-	57.4
	180	9	1.23	329.4	10	-	335.6	10	-	162.7
	190	6	2.03	458.3	10	-	592.4	10	-	293.9
0.5	150	10	-	87.0	10	-	43.3	10	-	18.7
	160	10	-	171.8	10	-	65.6	10	-	26.9
	170	10	-	264.8	10	-	107.4	10	-	40.2
	180	10	-	621.6	10	-	209.7	10	-	73.6
	190	3	2.09	763.7	10	-	329.9	10	-	111.6
0.7	150	10	-	203.1	10	-	40.1	10	-	11.6
	160	10	-	282.7	10	-	54.6	10	-	15.6
	170	10	-	498.3	10	-	77.8	10	-	24.5
	180	10	-	690.7	10	-	108.2	10	-	34.0
	190	5	1.11	934.3	10	-	188.4	10	-	51.4

Table 5 summarizes the results of our experiment on larger unweighted graphs. Note that the effect of our suggested improvements is clearly visible on this table. In particular, the total number of problem instances that could be solved to optimality by MMM_Benders is 68, while MMM_Benders_Improved was able to solve 172 instances to optimality. We also observe that optimality gaps for instances that could not be solved to optimality are reduced as a result of the improvements.

Finally, Table 6 compares MWMM_Benders and MWMM_Benders_Improved on weighted graphs having up to $|V| = 300$ vertices. Our observations on the results presented in Table 6 are similar to our previous observations. In particular: i) as $|V|$ increases, fewer instances can be solved to optimality and the average optimality gap tends to increase, ii) the performance of our approach increases as D increases for both cases, and iii) there is no significant difference between the performances of our decomposition algorithms for solving weighted and unweighted graphs having the same number of vertices.

7 Conclusions and Future Research

In this paper, we described exact solution algorithms for the problem of finding a minimum weight maximal matching in (edge) weighted graphs, and its unweighted version where each edge has unit weight. Our algorithms are based on Benders decomposition applied to appropriate reformulations of an integer programming formulation proposed in [31]. The master problem of our decomposition approach seeks an optimal vertex

Table 5: Comparison of algorithms for solving MMM on large graphs

		MMM_Benders			MMM_Benders_Improved		
D	$\|V\|$	Solved	Gap	Time	Solved	Gap	Time
	200	5	2.20	1091.8	10	-	443.3
	210	0	2.82	-	10	-	656.8
	220	0	3.38	-	2	1.00	1137.2
	230	0	4.35	-	0	2.38	-
	240	0	4.79	-	0	2.73	-
0.3	250	0	5.10	-	0	3.48	-
	260	0	5.63	-	0	4.17	-
	270	0	5.90	-	0	4.65	-
	280	0	6.44	-	0	4.70	-
	290	0	7.15	-	0	5.11	-
	300	0	7.34	-	0	5.51	-
	200	10	-	543.6	10	-	160.6
	210	10	-	701.4	10	-	207.0
	220	1	1.71	1045.3	10	-	291.5
	230	1	2.45	1067.6	10	-	422.5
	240	0	3.16	-	10	-	688.9
0.5	250	0	3.63	-	9	-	1044.6
	260	0	4.13	-	1	0.81	905.9
	270	0	4.35	-	0	1.64	-
	280	0	4.72	-	0	2.26	-
	290	0	4.63	-	0	2.83	-
	300	0	5.36	-	0	3.36	-
	200	10	-	270.0	10	-	77.5
	210	10	-	371.7	10	-	104.3
	220	10	-	596.0	10	-	167.0
	230	10	-	849.9	10	-	210.8
	240	1	1.06	1094.0	10	-	348.1
0.7	250	0	2.00	-	10	-	487.4
	260	0	2.48	-	10	-	634.0
	270	0	3.08	-	8	0.77	744.9
	280	0	3.26	-	9	0.74	941.4
	290	0	4.43	-	3	1.29	1055.0
	300	0	5.31	-	0	1.13	-

Table 6: Comparison of algorithms for solving MWMM on large graphs

		MWMM_Benders			MWMM_Benders_Improved		
D	$\|V\|$	Solved	Gap	Time	Solved	Gap	Time
	200	5	1.76	685.3	10	-	468.3
	210	2	2.61	632.8	7	1.40	778.3
	220	1	3.19	1178.1	5	1.40	979.8
	230	0	3.97	-	2	2.50	822.0
	240	0	4.52	-	0	3.17	-
0.3	250	0	5.35	-	0	3.81	-
	260	0	5.15	-	0	4.40	-
	270	0	5.89	-	0	4.72	-
	280	0	6.51	-	0	4.84	-
	290	0	7.28	-	0	5.25	-
	300	0	7.47	-	0	5.51	-
	200	10	-	537.7	10	-	169.7
	210	10	-	734.8	10	-	205.1
	220	1	1.71	1050.7	10	-	340.7
	230	1	2.45	1064.6	10	-	460.0
	240	0	3.25	-	10	-	717.8
0.5	250	0	3.63	-	8	0.85	1052.8
	260	0	4.04	-	1	0.99	865.1
	270	0	4.35	-	0	1.72	-
	280	0	4.72	-	0	2.18	-
	290	0	4.63	-	0	2.97	-
	300	0	5.36	-	0	3.50	-
	200	10	-	271.3	10	-	76.0
	210	10	-	371.4	10	-	105.6
	220	10	-	596.3	10	-	162.1
	230	10	-	849.0	10	-	219.5
	240	1	1.06	1098.2	10	-	345.9
0.7	250	0	2.00	-	10	-	419.7
	260	0	2.48	-	10	-	639.0
	270	0	3.08	-	9	0.77	764.2
	280	0	3.26	-	9	1.48	973.1
	290	0	4.43	-	3	1.23	1125.6
	300	0	5.31	-	0	1.66	-

cover, and the subproblem seeks a minimum weight perfect matching in the subgraph induced by the vertex cover. We showed how Benders feasibility cuts can be derived by using Gallai-Edmonds decomposition, and how Benders optimality cuts can be generated by solving the subproblem using a combinatorial matching algorithm. We tested the performance of our algorithms on randomly generated graph instances. Our results indicate that our decomposition approach clearly outperforms directly solving the underlying integer programming formulations for both problems.

As future research, one can consider developing exact methods for MMM and MWMM in some specific graph classes such as bipartite graphs and regular graphs by taking advantage of their structural properties. Also, developing a combinatorial branch-and-bound algorithm for the minimum weight maximal matching problem merits further research.

Acknowledgments. The authors are grateful to three anonymous referees and an associate editor, whose remarks helped improve the content and presentation of the paper.

References

[1] A. K. Andreas and J. C. Smith, Decomposition algorithms for the design of a nonsimultaneous capacitated evacuation tree network, Networks 53 (2009), 91–103.

[2] J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4 (1962), 238–252.

[3] A. Berger, T. Fukunaga, H. Nagamochi, and O. Parekh, Approximability of the capacitated b-edge dominating set problem, Theoretical Computer Science 385 (2007), 202–213.

[4] M. Chlebík and J. Chlebíková, Approximation hardness of edge dominating set problems, Journal of Combinatorial Optimization 11 (2006), 279–290.

[5] I. Contreras, J.-F. Cordeau, and G. Laporte, Benders decomposition for large-scale uncapacitated hub location, Operations Research 59 (2011), 1477–1490.

[6] A. M. Costa, A survey on Benders decomposition applied to fixed-charge network design problems, Computers & Operations Research 32 (2005), 1429–1450.

[7] R. S. de Camargo, G. Miranda Jr., and H. P. Luna, Benders decomposition for the uncapacitated multiple allocation hub location problem, Computers & Operations Research 35 (2008), 1047–1064.

[8] M. Demange and T. Ekim, Minimum maximal matching is NP-hard in regular bipartite graphs, TAMC 2008, Lecture Notes in Computer Science 4978 (2008), 364–374.

[9] M. Demange, T. Ekim, and C. Tanasescu, Hardness and approximation of minimum maximal matching, Manuscript submitted for publication, 2012.

[10] B. Dezsoa, A. Juttnerb, and P. Kovacs, LEMON – an open source C++ graph template library, Electronic Notes in Theoretical Computer Science 264 (2011), 23–45.

[11] J. Edmonds, Paths, trees and flowers, Canadian Journal of Mathematics 17 (1965), 449–467.

[12] F. V. Fomin, S. Gaspers, S. Saurabh, and A. A. Stepanov, On two techniques of combining branching and treewidth, Algorithmica 54 (2009), 181–207.

[13] T. Fujito, On approximability of the independent/connected edge dominating set problems, Information Processing Letters 79 (2001), 261–266.

[14] T. Fujito and H. Nagamochi, A 2-approximation algorithm for the minimum weight edge dominating set problem, Discrete Applied Mathematics 118 (2002), 199–207.

[15] M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman & Co., New York, 1979.

[16] Z. Gotthilf, M. Lewenstein, and E. Rainshmidt, A $(2-c\frac{logn}{n})$ approximation algorithm for the minimum maximal matching problem, Approximation and Online Algorithms, Lecture Notes in Computer Science 5426 (2009), 267–278.

[17] J. D. Horton and K. Kilakos, Minimum edge dominating sets, SIAM Journal on Discrete Mathematics 6 (1993), 375–387.

[18] S. F. Hwang and G. J. Chang, The edge domination problem, Discussiones Mathematicae Graph Theory 15 (1995), 51–57.

[19] B. Keller and G. Bayraksan, Disjunctive decomposition for two-stage stochastic mixed-binary programs with generalized upper bound constraints, INFORMS Journal on Computing 24 (2012), 172–186.

[20] G. Laporte and F. V. Louveaux, The integer L-shaped method for stochastic integer programs with complete recourse, Operations Research Letters 13 (1993), 133–142.

[21] G. Laporte and F. V. Louveaux, "Solving stochastic routing problems with the integer L-shaped method," Fleet Management and Logistics, T.G. Crainic and G. Laporte (Editors), Springer US, 1998, pp. 159–167.

[22] L. Lovász and M.D. Plummer, Matching theory, North-Holland, Amsterdam, 1986.

[23] Y. Matsumoto, N. Kamiyama, and K. Imai, An approximation algorithm dependent on edge-coloring number for minimum maximal matching problem, Information Processing Letters 111 (2011), 465–468.

[24] S. L. Mitchell and S. T. Hedetniemi, Edge domination in trees, Proc 8th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, 1977, pp 489–509.

[25] R. Montemanni, A Benders decomposition approach for the robust spanning tree problem with interval data, European Journal of Operational Research 174 (2006), 1479–1490.

[26] M. B. Richey and R. G. Parker, Minimum-maximal matching in series-parallel graphs, European Journal of Operations Research 33 (1988), 98–105.

[27] W. Romisch and S. Vigerske, "Recent progress in two-stage mixed-integer stochastic programming with applications to power production planning," Handbook of Power Systems I, P.M. Pardalos, S. Rebennack, M.V.F. Pereira, and N.A. Iliadis (Editors), Springer, Berlin, Germany, 2010, pp. 177–208.

[28] R. Schmieda and C. Viehmann, Approximating edge dominating set in dense graphs, Theoretical Computer Science 414 (2012), 92–99.

[29] A. Srinivasan, K. Madhukar, P. Nagavamsi, C. P. Rangan, and M.-S. Chang, Edge domination on bipartite permutation graphs and cotriangulated graphs, Information Processing Letters 56 (1995), 165–171.

[30] Z. C. Taşkın, "Benders decomposition," Encyclopedia of Operations Research and Management Science, J.J. Cochran (Editor), John Wiley & Sons, Malden, MA, 2010.

[31] Z. C. Taşkın and T. Ekim, Integer programming formulations for the minimum weighted maximal matching problem, Optimization Letters 6 (2012), 1161–1171.

[32] J. M. M. van Rooij and H. L. Bodlaender, Exact algorithms for edge domination, Parameterized and Exact Computation, Lecture Notes in Computer Science 5018 (2008), 214–225.

[33] M. Yannakakis and F. Gavril, Edge dominating sets in graphs, SIAM Journal on Applied Mathematics 38 (1980), 364–372.