Discrete Applied Mathematics 359 (2024) 143–152

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Polyhedral approach to weighted connected matchings in general graphs

Phillippe Samer [a,*], Phablo F.S. Moura [b]

[a] Universitetet i Bergen, Institutt for Informatikk, Postboks 7800, 5020, Bergen, Norway
[b] Research Center for Operations Research & Statistics, KU Leuven, Belgium

ARTICLE INFO

Article history:
Received 12 October 2023
Received in revised form 19 June 2024
Accepted 26 July 2024
Available online xxxx

Keywords:
Connected matchings
Combinatorial optimization
Polyhedral combinatorics
Integer programming
Induced connectivity
Connected subgraphs

ABSTRACT

A connected matching in a graph G consists of a set of pairwise disjoint edges whose covered vertices induce a connected subgraph of G. While finding a connected matching of maximum cardinality is a well-solved problem, it is NP-hard to determine an optimal connected matching in an edge-weighted graph, even in the planar bipartite case. We present two mixed integer programming formulations and a sophisticated branch-and-cut scheme to find weighted connected matchings in general graphs. The formulations explore different polyhedra associated to this problem, including strong valid inequalities both from the matching polytope and from the connected subgraph polytope. We conjecture that one attains a tight approximation of the convex hull of connected matchings using our strongest formulation, and report encouraging computational results over DIMACS Implementation Challenge benchmark instances. The source code of the complete implementation is also made available.

1. Introduction

A P-matching in a graph G consists of a matching M such that the subgraph induced by vertices covered by M has some property P, *e.g.* being connected. This paper is devoted to the problem of computing maximum weight *connected matchings* in a general graph: a set of pairwise disjoint edges of maximum total weight, whose covered vertices induce a connected subgraph of G.

Exciting results on the computational tractability of determining connected matchings attract justified attention in the literature around this appealing generalization of classical matchings, which are already such a fundamental structure in bridging theory and sophisticated applications across domains that range from early logistics, as the postperson problem illustrates, to novel programs in kidney paired exchange [23]. A striking dichotomy here is that finding a maximum cardinality connected matching is a well-solved problem, while the edge-weighted counterpart is NP-hard even in very restricted graph classes. As we outline below, our contributions represent a step forward in sharpening our ability to face that challenge, proposing a polyhedral combinatorics framework to actually determine maximum weight connected matchings in practice.

We remark that work on P-matching problems dates back at least to Stockmeyer and Vazirani [30] on induced matchings. Increased attention is due to thorough advances by Golumbic et al. [16] on uniquely restricted matchings, and Goddard et al. [14] contemplating acyclic, connected, and disconnected matchings. More recently, a number of fine-grained complexity results about the weighted connected matching (WCM) problem were presented by Gomes et al.

* Corresponding author.
E-mail addresses: samer@uib.no (P. Samer), phablo.moura@kuleuven.be (P.F.S. Moura).

https://doi.org/10.1016/j.dam.2024.07.042

[17,18]. They establish the NP-hardness of finding maximum weight connected matchings, for instance, even in planar graphs of maximum vertex degree 3 and edge weights in $\{-1, 1\}$, and in planar bipartite graphs with edge weights in $\{0, 1\}$.

In light of that complexity barrier, our hope is to bring the machinery of polyhedral studies and mixed-integer linear programming (MILP) to bear on the investigation of WCM in general graphs. We seek to contribute both to the theoretical study in understanding and approximating the polytope $\mathfrak{C}(G)$ of connected matchings (*i.e.* the convex hull of characteristic vectors in $\mathbb{R}^{|E(G)|}$ of connected matchings in G), and to practical algorithms and their computer implementation. We now stand on decades' worth of progress in matching theory, in combinatorial optimization problems around connectivity and network design, and in mathematical programming computation. It is therefore expected that we are able to harness the polyhedral point of view, and evaluate to what extent it leads to the practical solution of the WCM problem.

The main idea in this paper is that there are powerful, elegant polyhedral descriptions of WCM in general graphs, in the sense that we may expect a strong foundation of polyhedral results and progressively more effective MILP solvers for this problem. We defend the standpoint that only the combination of theoretical and applied results from communities in combinatorics and mathematical programming may truly settle (and push) the limitations around finding optimal weighted connected matchings. From this perspective, the carefully designed formulations and the open-source software that we propose are useful ingredients towards that end.

In summary, our main contributions are the following.

1. Polyhedral descriptions yielding exact integer programming algorithms to find weighted connected matchings in general graphs. We present both a compact, extended formulation that can be easily fed to a black-box solver, and an exponential formulation in the space of natural variables only, using blossom inequalities from the matching polytope, and minimal separators and indegree inequalities from the connected subgraph polytope.
2. Detailed presentation of a sophisticated branch-and-cut scheme based on the exponential formulation. The resulting algorithm, as well as the solver based on the compact formulation, attains encouraging computational performance on four different sets of benchmark instances of connected subgraph problems from the 11th DIMACS Implementation Challenge, and settles a state-of-the-art baseline for future work.
3. Free, open-source repository with the complete implementation, including a series of useful, general-purpose algorithmic components.

2. Polyhedral descriptions of weighted connected matchings

In this section we present the main idea of the paper. We concentrate here on the polyhedra leading to MILP formulations for WCM. Section 3 continues with algorithmic aspects, including our particular design choices based on preliminary computational evaluations.

Our terminology and notation are standard in algorithmics and graph theory. Note that we write $[k] \stackrel{\text{def}}{=} \{1, \ldots, k\}$, and that we denote by 2^S the power set of S, that is, the set of all subsets of S.

2.1. Extended formulation

We begin with a compact, extended formulation. That is, a system of inequalities in higher dimensional space which (i) has a number of variables and constraints that is polynomial in the input size, and (ii) whose orthogonal projection into the original space contains all (and only those) lattice points corresponding to integer feasible solutions. In particular, we use the well-known approach of modeling the flow of a commodity in an auxiliary network to impose the connectivity of the induced subgraph; see [25] for a thorough introduction.

We denote the flow network by $\mathcal{D} = (V(G) \cup \{s\}, \mathcal{A})$ where s is an artificial source vertex, and $\mathcal{A}$ contains both orientations of each original edge in G, besides an arc from s to each other vertex. That is, $\mathcal{A} \stackrel{\text{def}}{=} \{(u, v), (v, u) : \text{ for } \{u, v\} \in E(G)\} \cup \{(s, u) : u \in V(G)\}$. As usual, let $n \stackrel{\text{def}}{=} |V(G)|$, $m \stackrel{\text{def}}{=} |E(G)|$, $\delta : V(G) \to 2^E$ denote the set of edges incident to a vertex of graph G, and let $\delta^+, \delta^- : V(\mathcal{D}) \to 2^{\mathcal{A}}$ denote the set of arcs leaving (resp. entering) a vertex of network $\mathcal{D}$.

The model we propose imposes connectivity of the solution by requiring that there be an arborescence rooted in s, so that there is an arc reaching a given vertex if and only if it is saturated by a matching edge. To accomplish that, we note that each matching M covers $2 \cdot |M|$ vertices, and so determine that exactly $2 \cdot |M|$ *units of flow* leave the artificial source s, and that each covered vertex *absorbs* a flow unit. Specifically, a first MILP formulation of the WCM problem is given by

$$\max \left\{ \sum_{e \in E(G)} w_e x_e : (\mathbf{x}, \mathbf{y}, \mathbf{f}) \in \mathcal{P}_{\text{ext}}(G) \cap \{0, 1\}^m \times \{0, 1\}^{2m+n} \times \mathbb{Q}_+^{2m+n} \right\}, \tag{1}$$

where $\mathcal{P}_{\text{ext}}(G)$ is the following polyhedral region:

$$\sum_{e \in \delta(u)} x_e \leq 1 \qquad \text{for each } u \in V(G), \tag{2}$$

$$\sum_{a\in\delta^-(u)} y_a = \sum_{e\in\delta(u)} x_e \qquad \text{for each } u \in V(G), \tag{3}$$

$$\sum_{u\in V(G)} y_{su} \leq 1, \tag{4}$$

$$y_{uv} \leq \sum_{a\in\delta^-(u)} y_a \qquad \text{for each } u \in V(G) \text{ and each } uv \in \delta^+(u), \tag{5}$$

$$f_a \leq n \cdot y_a \qquad \text{for each } a \in \mathcal{A}, \tag{6}$$

$$\sum_{u\in V(G)} f_{su} = 2 \cdot \sum_{e\in E(G)} x_e, \tag{7}$$

$$\sum_{a\in\delta^-(u)} f_a - \sum_{a\in\delta^+(u)} f_a = \sum_{a\in\delta^-(u)} y_a \qquad \text{for each } u \in V(G), \tag{8}$$

$$x_e \geq 0 \qquad \text{for each } e \in E(G), \tag{9}$$

$$y_a \geq 0 \qquad \text{for each } a \in \mathcal{A}, \tag{10}$$

$$f_a \geq 0 \qquad \text{for each } a \in \mathcal{A}. \tag{11}$$

Note that variables $\mathbf{x}$ determine which edges of G are in the solution matching, variables $\mathbf{y}$ determine an orientation by allowing arcs of $\mathcal{D}$ to carry flow, and variables $\mathbf{f}$ give the actual flow running on each arc.

The classical degree inequalities in (2) are enough to have integer points where at most one edge reaches each vertex. Constraints (3) link $\mathbf{x}$ and $\mathbf{y}$ variables, by setting the number of (directed) arcs entering u as the number of (undirected) edges incident to it — either zero or one, as enforced by the previous constraints. Constraint (4) establishes that the artificial flow source s should be linked to at most one vertex in G; note that we expect exactly one arc leaving s in interesting examples, but the model still allows an empty solution (*e.g.* when the objective coefficients $\mathbf{w}$ are negative everywhere). Constraints (5), which capture that we may only open an out arc from u to v if some arc arrives at u, are actually implied by the other sets of inequalities but generally perceived as helping solve LP relaxation faster.

The remaining constraints concern the network flow. Inequalities (6) bind $\mathbf{y}$ and $\mathbf{f}$ variables: nonzero flow is only allowed in open arcs, and the maximum flow is $n = |V(G)|$. Constraint (7) establishes that the flow leaving the artificial source s is exactly the number of vertices saturated by the matching (*i.e.* twice as many as there are edges in the matching). Lastly, flow balance constraints (8) impose a single connected component in the solution: vertices in the arborescence (namely, those whose number of incoming arcs in the right-hand side is one) consume one unit of flow, while others may not interfere either consuming or creating flow.

The main advantage of having a MILP formulation with polynomial number of variables and constraints is the practicality of just feeding it to a black-box solver, automatically benefiting of increased performance due to software and hardware improvement. On the other hand, while an extended formulation may have much smaller number of facets than its projection, decades of mathematical programming computation led to numerous examples where superior performance is attained by branch-and-cut algorithms that dynamically identify and add cutting-planes violated by relaxation solutions. That is the path we now thread. First, designing a formulation for WCM anchored in solid knowledge of the underlying connected subgraph polytope and the classical matching polytope. Next, in Section 3, filling in the details of the best-performing branch-and-cut scheme we devised and offer in our accompanying software repository.

2.2. Formulation in the original space of variables

The guiding principle in the design of our second formulation for WCM is to waive the concern about model size and build on strong valid inequalities leading to the best-performing solvers for closely related problems, defined over the larger polytopes of classical matchings and connected subgraphs.

The classical matching polytope is well-known since the very birth of the polyhedral combinatorics field — tied to the celebrated results of Edmonds [10], and better understood in light of the combinatorial proof of Balinski [3]. Namely, Edmonds showed that, together with degree inequalities in (2) above, *blossom inequalities* give all the facets missing in a complete characterization of the matching polytope, and can be separated efficiently.

On the other hand, as the maximum-weight connected subgraph (MWCS) problem is NP-hard even in very restricted particular cases [21], there is no hope for an ideal formulation of the connected subgraph polytope under the assumption that P $\neq$ NP and the equivalence of separation and optimization. While there are many options for modeling induced connectivity, a recent performance breakthrough in exact solvers for problems like the MWCS [1,31] and Steiner trees [11] is attributed to *minimal separator inequalities* (MSI) on vertex choosing variables $y \in \{0, 1\}^{|V(G)|}$:

$$y_a + y_b - \sum_{u\in S} y_u \leq 1, \tag{12}$$

for each pair of non-adjacent vertices a and b, and each (a, b)-separator $S \subseteq V \setminus \{a, b\}$, *i.e.* there are no paths connecting a to b if we remove S from G. The eminently readable paper by Wang et al. [31] includes a proof that (12) defines a facet of the connected subgraph polytope if and only if the separator S is minimal.

It is worth remarking that the breakthrough we refer to in practical evidence (runtime of resulting MILP solvers) does not agree with the theoretical intuition given by inclusion of different polyhedral relaxations. In particular, the recent work of Rehfeldt et al. [28] proves that the LP bound from the MSI relaxation for induced connectivity is weaker than earlier alternatives based on combining vertex and edge-variables. That is in stark contrast to the experimental results mentioned above. In particular, the praised solver of Fischetti et al. [11] won most of the categories at the 11th DIMACS Implementation Challenge [9]. In line with our guiding principle in this section, we follow the standpoint of those authors, who conclude that a simpler model defined in the natural space of variables and a careful implementation framework enabled their superior performance.

To define a system of inequalities combining the separation of MSI (12) and only using natural design variables $\mathbf{x} \in \{0, 1\}^{|E(G)|}$ in the original space of polytope $\mathfrak{C}(G)$, we use the fact that vertex u belongs to the subgraph induced by matching M if and only if there is exactly one edge in M incident to u. Hence, projecting the MSI onto the space of $\mathbf{x}$ variables using $y_u \mapsto \sum_{e \in \delta(u)} x_e$, we derive the first IP formulation to find maximum weight connected matchings using MSI,

$$\max \left\{ \sum_{e \in E(G)} w_e x_e : \mathbf{x} \in \mathcal{P}_{\text{sep}}(G) \cap \{0, 1\}^m \right\}, \tag{13}$$

with $\mathcal{P}_{\text{sep}}(G)$ defined by degree inequalities (2), non-negativity bounds (9), and the projection of separator inequalities (12):

$$\sum_{e \in \delta(a)} x_e + \sum_{e \in \delta(b)} x_e - \sum_{u \in S} \sum_{e \in \delta(u)} x_e \leq 1 \qquad \text{for each } a, b \in V(G), \{a, b\} \notin E(G), \text{ and each } S \in \mathcal{C}(a, b),$$

where $\mathcal{C}(a, b) \subset 2^{V(G)}$ denotes the set of all minimal (a, b)-separators in G.

We reinforce this formulation with two exponential families of valid inequalities. The first is an additional class of facets of the connected subgraph polytope studied by Wang et al. [31]. The other was already mentioned earlier: blossom inequalities, which define all remaining facets of the classical matching polytope.

Indegree inequalities A vector $d \in \mathbb{Z}_+^n$ is called an *indegree* vector of graph G if there exists an orientation of its edges such that the indegree of each vertex u is d_u. Introduced decades earlier in the context of greedoid optimization and only in the particular case where G is a tree [22, Chapter XI, Theorem 3.6], it was later shown by Wang et al. [31] that, for each indegree vector d of an *arbitrary* graph G, inequality

$$\sum_{u \in V(G)} (1 - d_u) \cdot y_u \leq 1$$

is valid for the connected subgraph polytope of G.

Interestingly, the indegree inequalities provide a minimal, complete characterization of that polytope when G is a tree: each indegree inequality defines a facet, and none is missing. More importantly in the context of our problem, Wang et al. [31] prove that the indegree inequalities can still define facets in the general case, and may be separated in time proportional to $O(n + m)$. Again, we shall use the projection of those inequalities in the original space of the connected matching polytope $\mathfrak{C}(G)$ by the linear map $y_u \mapsto \sum_{e \in \delta(u)} x_e$.

Blossom inequalities Finally, to ensure that our formulation is within the tightest possible description of the (classical) matching relaxation, one would naturally expect the inclusion of blossom inequalities. Namely, for each *handle* $H \subset V(G)$ of odd cardinality, the inequality

$$\sum_{e \in E(G[H])} x_e \leq \frac{|H| - 1}{2}$$

is valid for the matching polytope of G. Besides being sufficient to determine the convex hull of incidence vectors of matchings in an arbitrary graph, each blossom inequality is also necessary when G is a complete graph, for example. They are also an important ingredient in state-of-the-art solvers for the traveling salesperson problem [7, Section 7.4].

Putting the inequalities together, the complete formulation on which we base our enhanced branch-and-cut algorithm to find weighted connected matchings is

$$\max \left\{ \sum_{e \in E(G)} w_e x_e : \mathbf{x} \in \mathcal{P}_{\text{full}}(G) \cap \{0, 1\}^m \right\}, \tag{14}$$

where $\mathcal{P}_{\text{full}}(G)$ is the following polyhedral region:

$$\sum_{e\in\delta(u)} x_e \le 1 \qquad \text{for each } u \in V(G), \tag{15}$$

$$\sum_{e\in\delta(a)} x_e + \sum_{e\in\delta(b)} x_e - \sum_{u\in S}\sum_{e\in\delta(u)} x_e \le 1 \qquad \text{for each non-adjacent } a, b \in V(G), \text{ and each } S \in \mathcal{C}(a,b), \tag{16}$$

$$\sum_{u\in V(G)} (1-d_u) \sum_{e\in\delta(u)} x_e \le 1 \qquad \text{for each indegree vector } d \text{ of } G, \tag{17}$$

$$\sum_{e\in E(G[H])} x_e \le \frac{|H|-1}{2} \qquad \text{for each } H \subset V(G) \text{ with } |H| \text{ odd}, \tag{18}$$

$$x_e \ge 0 \qquad \text{for each } e \in E(G). \tag{19}$$

3. Branch-and-cut scheme for the exponential formulation

The enhanced WCM formulation (14) is only useful in practice if an implementation of efficient separation procedures for the (exponentially-many) inequalities in (16), (17) and (18) defining $\mathcal{P}_{\text{full}}(G)$ is available. This section completes our main contribution, filling in the algorithmic details and presenting our free, open-source software package with the resulting solver for WCM in general graphs.

We designed our C++ code with attention to time and space efficiency, and fairly tested it for correctness along months of development. It is available in the wcm-branch-and-cut repository on GitHub (https://github.com/phillippesamer/wcm-branch-and-cut), thus welcoming collaboration towards extensions and facilitating the direct comparison with eventual algorithms designed for the WCM problem in the future. Moreover, the code can be forked and turned into useful algorithmic components to different problems and applications.

3.1. Separation procedures

Efficient separation algorithms are at the core of a successful branch-and-cut scheme, as they are executed a number of times in each node of the enumeration tree partitioning the search space. Since the classes of inequalities (16), (17) and (18) grow exponentially with the size of the input graph, it is not practical to add them in an explicit model *a priori* for reasonably sized problems. Except for the last method presented below, the following are *exact* separation procedures: oracles that, given a relaxation solution $\mathbf{x}^*$, either identify a specific inequality valid for $\mathcal{P}_{\text{full}}(G)$ that is violated at $\mathbf{x}^*$ (which can then be added to the formulation and *cut off* $\mathbf{x}^*$ to continue the search), or certify implicitly that no such inequality exists. In contrast, when it comes to blossom inequalities (18), we use both an exact and a *heuristic* separation procedure, *i.e.* a faster method to search for a blossom cut, that may fail even when one exists.

MSI cuts

We followed the description of Fischetti et al. [11, Section 2.1] for two exact separation algorithms for MSI in the award-winning solver mentioned in Section 2.2.

The first algorithm is based on the computation of maximum flows in a support digraph D, whose arc capacities are defined according to the current relaxation solution. For each pair of non-adjacent vertices $u, v \in V(G)$ such that $x^*_u + x^*_v > 1$ (which is necessary for an MSI to be violated), we calculate a maximum (u, v)-flow $\bar{f}$ in D. If $\bar{f} < x^*_u + x^*_v - 1$, we may determine a violated separator inequality from the corresponding minimum cut. Two implementation tweaks are worth mentioning.

(A) As first observed by Miyazawa et al. [26, Section 3.1], we often have a large number of variables in the relaxation solution $\mathbf{x}^*$ at either 0 or 1, and none of these appear in a (u, v)-separator that gives a violated inequality. We may therefore contract the corresponding arcs and vertices in D to run the separation algorithm in considerably smaller support digraphs. Implementing such contractions requires special care to keep track of the original variables that make up the violated inequality we eventually find.

(B) When we identify a minimum cut C yielding a violated (u, v)-separator inequality, it is in general not a *minimal* separator. As mentioned in Section 2.2, we thus have the opportunity to *lift* the left-hand side towards a minimal separator $S \subset C$ to derive a non-dominated inequality. This can be achieved with a simple graph traversal procedure, as formalized by Fischetti et al. [11, Section 2.1]. While they refer to a breadth-first search (BFS) and use an edge-deletion operation, we observe faster runtimes combining (i) a depth-first search that avoids an explicit BFS queue, and (ii) a boolean mask of *active*/*inactive* edges passed as a reference parameter instead of modifying the graph.

The worst-case time complexity of the whole procedure is in $O(n^2 \cdot g(\tilde{n}, \tilde{m}))$, where $n \stackrel{\text{def}}{=} |V(G)|$, $m \stackrel{\text{def}}{=} |E(G)|$, and $g(\tilde{n}, \tilde{m})$ denotes the complexity of a single maximum-flow computation in a digraph with $\tilde{n}$ vertices and $\tilde{m}$ arcs. We use the highly tuned implementation of the preflow-push algorithm of Goldberg and Tarjan [15] in COIN-OR LEMON 1.3.1 — the

Library for Efficient Modeling and Optimization in Networks [8], as the responsible team reports that best computational runtimes are attained with that algorithm. Its time complexity $g(\tilde{n}, \tilde{m})$ is in $O(\tilde{n}^2\sqrt{\tilde{m}})$. Since the digraph D above is such that $\tilde{n} \leq 2n$ and $\tilde{m} \leq 2m+n$ before the contractions due to integer-valued variables, the runtime of our separation is within $O(n^4\sqrt{n+m})$. Contrary to what one could expect from such a high worst-case time complexity, we had very encouraging numerical results in practice, which we attribute to the digraph contractions and the particular branch-and-cut scheme that we outline in Section 3.2 below.

An alternative, more efficient separation procedure is readily available in the particular case where the relaxation solution $\mathbf{x}^*$ is actually integer-valued. We may resort to a simple depth-first search (DFS) to check connectivity of the induced solution. In a disconnected solution, inspecting the neighborhood C of any connected component with a vertex u such that $x^*_u = 1$ gives a violating (u, v)-separator, for some v in a different component with $x^*_v = 1$. It is important to stress that implementation tweak (B) above applies here as well, and that we still need to derive a minimal separator $S \subset C$ to add a stronger inequality. The time complexity of the separation in this particular case is dominated by the DFS step, and is thus in $O(n + m)$.

It is fair to remark that MSI first appeared in two, earlier papers in applied operations research. Fügenschuh and Fügenschuh [12] introduced this class of inequalities and their separation algorithm in an intricate, non-linear programming problem arising in the sheet metal industry. Their work was picked up by Carvajal et al. [6], who extended on the role of MSI when imposing connectivity in a forestry planning problem. The MSI were also introduced independently in the polyhedral studies of the convex recoloring problem [5]. We also praise, again, the thorough, instructive chapter of Álvarez-Miranda et al. [1] on the maximum weight connected subgraph problem more generally.

Indegree cuts

The separation of indegree inequalities is remarkably simple. We implemented the procedure exactly as presented by Wang et al. [31]. The main point is to consider an orientation maximizing the left-hand side over all indegree vectors, namely: taking $\overrightarrow{uv}$ for $\{u, v\} \in E(G)$ if and only if $x^*_u \geq x^*_v$. The worst-case time complexity of the algorithm is in $O(n+m)$.

Blossom cuts

We consider two separation algorithms for blossom inequalities. See Section 3.2 below for the detailed scheme in which they are used.

The first method is the exact separation procedure of Padberg and Rao [27]. We strictly followed its presentation *cf.* [24], who introduced the state-of-the-art algorithm for the more general version of b-matchings with edge capacities. The separation works on a support graph G' with $n + 1$ vertices and $m + n$ edges, whose capacities are determined by the current relaxation solution $\mathbf{x}^*$. It boils down to determining a *cut tree* $\mathcal{T}(G')$: an elegant data structure introduced by Gomory and Hu [19] to encode minimum cuts between all pairs of vertices in the graph, at the expense of computing only $|V(G')| - 1$ maximum flow computations.

We use COIN-OR LEMON 1.3.1 [8] here as well to build the cut tree $\mathcal{T}(G')$. Then, it suffices to verify the blossom inequality $\sum_{e\in E(G[H])} x^*_e \leq {}^{(|H|-1)}/_2$ for at most one handle $H \subset V(G)$ per edge of the tree. Constructing the data structure dominates the worst-case time complexity of the complete algorithm, and is within $O(n^4)$ [24]. We remark that an implementation here may construct the support graph G' only once, and just update edge capacities according to the current relaxation values.

Our second method is inspired by the work of Bicalho et al. [4, Section 4.1.2], who observed comparable results using the exact method above and a separation heuristic for blossom inequalities in a row-and-column generation algorithm for a different network design problem. We devised a simpler algorithm to try to find blossom cuts in linear time as follows. Let H denote the support graph induced only from fractional variables in $\mathbf{x}^*$, and let H_i denote the connected components of H. For each H_i of odd cardinality, we simply inspect the corresponding blossom inequality for violation. The complexity of this separation heuristic is dominated by the step finding connected components in H, which is in $O(n+m)$ in the worst case.

3.2. Further algorithmic aspects

We are now ready to depict our complete branch-and-cut scheme. We use the overall framework in the Gurobi 10.0.2 solver [20], with callbacks to implement separation procedures. It is important to distinguish between the specific callback from a new MIP incumbent, where only *lazy constraints* are added (in our case, MSI tailored for integer-valued points), and the standard callback from the search at a given MIP node, where we add *user cuts* (all of MSI, indegree, and blossom inequalities).

In the beginning, only degree inequalities (15) are included *a priori* in the model. Instead of relying on the solver standard behavior concerning how long to explore the root node relaxation before branching, we designed a **strengthened root node LP relaxation**. Here, we dedicate up to 300 s or 10% of the specified time limit (whichever is shorter) prior to the main call to the solver, and alternate the solution of an LP relaxation and cut generation observing that:

1. All MSI violated in the current relaxation solution $\mathbf{x}^*$ are added;
2. The exact separation of blossom inequalities is attempted only if $\mathbf{x}^*$ is fractional, no MSI cut was found and the separation heuristic failed;

3. No indegree cuts are added unless all other separation algorithms failed, to prevent the inclusion of an excessively large number of constraints.

The reinforced model resulting from this initialization consistently showed the best computational performance across a range of configurations in our preliminary computational evaluation. In particular, we explain in Section 4.2 that a number of instances could be solved without resorting to branching — which was not the case before we devised this strengthened root node relaxation.

The general algorithm continues with the branch-and-cut enumeration tree, partitioning the search space by fixing a single binary variable, $x_u = 0$ or $x_u = 1$, in each branch. In each remaining node below the root relaxation, more attention is paid to limit the complexity of cut generation, by observing the following:

1. The MSI separation algorithm concludes as soon as any inequality violated at the current relaxation is found, iterating over the initial source vertex considered for maximum flow computations;
2. The exact separation of blossom inequalities is turned off, and only the heuristic is run;
3. The separation of indegree inequalities is turned off.

4. Experimental evaluation

We conclude our work with the first experimental evaluation of an algorithm for the weighted connected matching (WCM) problem. The main goal here is to indicate that our polyhedral descriptions and the resulting algorithms constitute a practical approach to find optimal connected matchings in non-trivial inputs.

4.1. Benchmark design

We hope to set a judicious baseline towards progress in the computation of WCM. Namely, one that is anchored in the *reproducibility* of materials and methods, and that reports experimental evidence from respectable, interesting testbeds.

Interlude

At an early stage of our experiments, we considered using binomial (Erdős-Rényi) graphs $\mathcal{G}_{n,p}$ as reported by Wang et al. [31] when studying the performance of minimal separator and indegree inequalities for the maximum weight connected subgraph (MWCS) problem. To our surprise, the random graphs in this model resulted in quite simple WCM instances. Both the compact extended formulation and the exponential one in the original space give so tight bounds that problems in the order of 10^5 vertices were solved in negligible time on a desktop computer — whether on sparse or dense $\mathcal{G}_{n,p}$ graphs ($p \in [0.01, 0.6]$), whether with Gaussian or uniform random weights. The script to produce such instances using the robust NetworkX Python package is still available in our GitHub repository. Nevertheless, we do not go further in evaluating those examples in our research. Instead, we choose to proceed by borrowing credibility from a certified source of benchmark instances of MWCS and similar problems.

Benchmark instances

Our computational evaluation is carried out over benchmark instances from the 11th DIMACS/ICERM Implementation Challenge. The competition covered several variants around the Steiner tree problem, and we chose to use all three sets of instances of the MWCS problem, and the one set available for the Generalized Maximum-Weight Connected Subgraph (GMWCS) problem. Specifically, there are 39 instances in set MWCS-GAM, 72 in MWCS-JMPALMK, 8 in MWCS-ACTMOD, and 63 instances in set GMWCS-GAM. Table 1 in the supplementary material (Appendix A) contains the full names, sizes, and a numerical ID for ease of reference of the 182 instances. The smallest instances have less than a thousand vertices and edges; the largest ones exceed 5.000 vertices and 90.000 edges. See [9] for more information.

Since MWCS instances contain only vertex weights, we determined $w(e) \stackrel{\text{def}}{=} w(u) + w(v)$ for each $e = \{u, v\} \in E(G)$. For GMWCS problems we used the edge weights included in the input instance, and ignored vertex weights. We note that 10 out of the 63 GMWCS instances have only negative weights, and so the resulting WCM problem has null optimum. We decided to keep those instances in the benchmark for the sake of completeness.

Platform and settings

We tested the implementation on a desktop machine with an Intel Core i5-8400 processor, with 6 CPU cores at 2.80 GHz, and 16 GB of RAM, runnning GNU/Linux kernel 6.2.0–33 under the Ubuntu 22.04.3 LTS distribution. All the code is compiled with g++ 11.4.0.

As mentioned in Section 3.2, we use the Gurobi Optimization [20] MILP solver. We set a time limit of 3600 s in all executions, while noting that the solver process may exceed that by a negligible amount. All solver parameters are used in their default values, except for setting an extra effort on MIP heuristics when using the exponential formulation with our branch-and-cut scheme (`MIPFocus = 1` and `GRB_DoubleParam_Heuristics = 0.2`). In our implementation, we require a violation by at least 10^{-5} to add a cutting plane in all separation procedures.

4.2. Discussion

Table 2 in the supplementary material (Appendix A) contains the detailed results of the solver using the extended formulation (1), *i.e.* optimizing over polyhedron $\mathcal{P}_{\text{ext}}$, and our enhanced branch-and-cut scheme with formulation (14), which is based on the exponential polyhedral description in $\mathcal{P}_{\text{full}}$. We include information both on the LP relaxation and the full integer program in the table, and discuss our findings next.

Overall performance. A first note is about the actual practicality of the implementations. We were satisfied that the enhanced branch-and-cut scheme over $\mathcal{P}_{\text{full}}$ concludes with an optimality certificate for 168 out of 182 instances. The corresponding number for the compact formulation is 151 cases. Taking into account that we use at most one hour of processing by a regular desktop computer, we consider this a rather encouraging conclusion. Practitioners and applications with a connected matching subproblem should therefore be able to derive improved runtimes by taking advantage of more powerful computing platforms.

Empirical approximation of the ideal formulation. Concerning how tightly we approximate the convex hull $\mathfrak{C}(G)$ with our formulations, it is remarkable that the LP relaxation bound of $\mathcal{P}_{\text{full}}$ matches the optimum in 98 out of 168 instances for which the optimum is known, and the enhanced branch-and-cut algorithm is able to prove optimality in the root node relaxation in 111 problems. More generally, the optimum is within 5% of LP bound in 145 of those 168 instances. We observe that the LP relaxation bound of $\mathcal{P}_{\text{full}}$ is stronger or equal to that of $\mathcal{P}_{\text{ext}}$ in all instances. It is 23.9% tighter on average, and up to 84% tighter (recall that we impose a time limit for the LP relaxation, so it could be even stronger).

Comparing bounds between formulations. As expected, the dual bounds attained with the enhanced branch-and-cut algorithm over $\mathcal{P}_{\text{full}}$ are consistently stronger than that of the compact extended formulation, and there is only a single case where the latter is stronger (namely, the instance of ID 16). Concerning primal bounds, we were surprised positively with 8 examples where the compact extended formulation does find an integer feasible solution better than the exponential one. We refer to the results on instances with identifiers 3, 16, 21, 35, 36, 42, 44, 45.

Superiority of exponential formulation in harder instances. Finally, seeking a classification of the instances with respect to computational hardness, we find that 123 out of 182 instances could be solved to optimality by both formulations within 5 min. In the remaining 59 instances, we find good clues of the superiority of $\mathcal{P}_{\text{full}}$.

i. In 30 of those 59 cases, the exponential formulation does finish within 5 min.
ii. The LP relaxation bound of $\mathcal{P}_{\text{full}}$ is up to 65.0% stronger than that of $\mathcal{P}_{\text{ext}}$ in this subset; it is 34.7% stronger on average.
iii. In 11 instances, the exponential formulation solves the problem at the root relaxation node, whereas the compact one struggles to finish: not even proving optimality before 3600 s in 3 cases; taking 1510 or 2388 s in 2 cases; and 6 other cases taking longer than 5 min.
iv. While there are 14 instances where the solver with the exponential formulation could not prove optimality (*i.e.* exceeds the one hour time limit with an open gap), there are 31 such cases for the compact formulation.

5. Final remarks

The standing argument behind our work is that polyhedra and MILP should lead to an interesting, useful perspective to study WCM both in theory and in practice. This complements other methodologies that are currently available to find weighted connected matchings, which assume restricted input graph classes. Moreover, we hope that our approach also determines a solid baseline of comparison for further research crafting WCM algorithms.

Besides their appealing polyhedral structures and intrinsic connections with established problems in combinatorics and optimization, both formulations considered here had encouraging computational performance, and could be considered for eventual applications of the WCM problem as well. On the one hand, having good results from the compact extended formulation is an achievement in its own right (we remark that it was able to find better primal feasible solutions in a few examples), as performance improvements in the underlying MILP solver usually leads to better runtimes in such "simpler" models automatically.

Still, all the work in designing an enhanced formulation did pay off, and we are proud to contribute yet another success story where the theoretical insight gathered from careful polyhedral relaxations translates to strides in practical computing experience. Using the exponential description and the resulting branch-and-cut scheme, we provide optimality certificates for 168 out of 182 instances. Most noticeable, that formulation solves 111 out of 182 instances in the root relaxation, without resorting to branching.

We believe that further research should consider only the subset of harder instances discussed here, and investigate features that characterize the most challenging ones before proposing new benchmark sets. It should also be possible to strengthen the compact extended polyhedron as well, so that more instances could be solved to proven optimality within limited runtimes.

Finally, our software repository includes not only the implementation of all the methods presented in this paper, but also a simple tool using `polymake` [2,13] to assist one in inspecting the connected matching polytope and finding new classes of strong valid inequalities. We had some progress in this direction [29], and trust that many fruitful results could be derived by further research translating the polyhedral insight to improved WCM algorithms.

Data availability

The free, open-source code of all algorithms implemented, as well as the benchmark instances, is permanently available in the GitHub repository mentioned in the paper.

Acknowledgments

P. Samer gratefully acknowledges the work of the institute administration at UiB, according to the Working Environment Act §14-7 of the Royal Norwegian Ministry of Labour and Social Inclusion, which enabled the mobility period leading to the research results presented in this paper, as well as the support by the Research Council of Norway through the research project 249994 CLASSIS. P.F.S. Moura is supported by Internal Funds of KU Leuven, and partially supported by CNPq-Brazil (Proc. 404315/2023-2).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.dam.2024.07.042.

References

[1] E. Álvarez-Miranda, I. Ljubić, P. Mutzel, The maximum weight connected subgraph problem, in: M. Jünger, G. Reinelt (Eds.), Facets of Combinatorial Optimization: Festschrift for Martin Grötschel, Springer Berlin Heidelberg, 2013, pp. 245–270, URL http://dx.doi.org/10.1007/978-3-642-38189-8_11.
[2] B. Assarf, E. Gawrilow, K. Herr, M. Joswig, B. Lorenz, A. Paffenholz, T. Rehn, Computing convex hulls and counting integer points with polymake, Math. Program. Comput. 9 (2017) 1–38, URL http://dx.doi.org/10.1007/s12532-016-0104-z.
[3] M.L. Balinski, Establishing the matching polytope, J. Combin. Theory Ser. B 13 (1) (1972) 1–13, URL http://dx.doi.org/10.1016/0095-8956(72)90002-0.
[4] L.H. Bicalho, A.S. Da Cunha, A. Lucena, Branch-and-cut-and-price algorithms for the degree constrained minimum spanning tree problem, Comput. Optim. Appl. 63 (2016) 755–792, URL http://dx.doi.org/10.1007/s10589-015-9788-7.
[5] M. Campêlo, K.R. Lima, P.F. Moura, Y. Wakabayashi, Polyhedral studies on the convex recoloring problem, Electron. Notes Discrete Math. (ISSN: 1571-0653) 44 (2013) 233–238, URL http://dx.doi.org/10.1016/j.endm.2013.10.036.
[6] R. Carvajal, M. Constantino, M. Goycoolea, J.P. Vielma, A. Weintraub, Imposing connectivity constraints in forest planning models, Oper. Res. 61 (4) (2013) 824–836, URL http://dx.doi.org/10.1287/opre.2013.1183.
[7] M. Conforti, G. Cornuéjols, G. Zambelli, Integer Programming, Springer, Cham, 2014, URL http://dx.doi.org/10.1007/978-3-319-11008-0.
[8] B. Dezső, A. Jüttner, P. Kovács, LEMON – an open source C++ graph template library, Electron. Notes Theor. Comput. Sci. 264 (5) (2011) 23–45, URL http://dx.doi.org/10.1016/j.entcs.2011.06.003.
[9] DIMACS'11, in: David S. Johnson, Thorsten Koch, Renato F. Werneck, Martin Zachariasen (Eds.), The Eleventh DIMACS Implementation Challenge in Collaboration with ICERM: Steiner Tree Problems, 2014, URL https://dimacs11.zib.de/downloads.html.
[10] J. Edmonds, Maximum matching and a polyhedron with 0 1-vertices, J. Res. Natl. Bureau Stand. B 69 (125–130) (1965) 55–56, URL http://dx.doi.org/10.6028/jres.069B.013.
[11] M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, M. Sinnl, Thinning out steiner trees: a node-based model for uniform edge costs, Math. Program. Comput. 9 (2) (2017) 203–229, URL http://dx.doi.org/10.1007/s12532-016-0111-0.
[12] A. Fügenschuh, M. Fügenschuh, Integer linear programming models for topology optimization in sheet metal design, Math. Methods Oper. Res. 68 (2008) 313–331, URL http://dx.doi.org/10.1007/s00186-008-0223-z.
[13] E. Gawrilow, M. Joswig, Polymake: a framework for analyzing convex polytopes, in: G. Kalai, G.M. Ziegler (Eds.), Polytopes – Combinatorics and Computation, Springer, 2000, pp. 43–73, URL http://dx.doi.org/10.1007/978-3-0348-8438-9_2.
[14] W. Goddard, S.M. Hedetniemi, S.T. Hedetniemi, R. Laskar, Generalized subgraph-restricted matchings in graphs, Discrete Math. (ISSN: 0012-365X) 293 (1) (2005) 129–138, URL http://dx.doi.org/10.1016/j.disc.2004.08.027. 19th British Combinatorial Conference.
[15] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum-flow problem, J. ACM 35 (4) (1988) 921–940, URL http://dx.doi.org/10.1145/48014.61051.
[16] M.C. Golumbic, T. Hirst, M. Lewenstein, Uniquely restricted matchings, Algorithmica 31 (2001) 139–154, URL http://dx.doi.org/10.1007/s00453-001-0004-z.
[17] G.C. Gomes, B.P. Masquio, P.E. Pinto, V.F. dos Santos, Szwarcfiter. J.L., Disconnected matchings, Theoret. Comput. Sci. (ISSN: 0304-3975) 956 (2023) 113821, URL http://dx.doi.org/10.1016/j.tcs.2023.113821.
[18] G.C.M. Gomes, B.P. Masquio, P.E.D. Pinto, V.F.d. Santos, J.L. Szwarcfiter, Weighted connected matchings, in: A. Castañeda, F. Rodríguez-Henríquez (Eds.), LATIN 2022: Theoretical Informatics, Springer International Publishing., Cham, ISBN: 978-3-031-20624-5, 2022, pp. 54–70, URL http://dx.doi.org/10.1007/978-3-031-20624-5_4.
[19] R.E. Gomory, T.C. Hu, Multi-terminal network flows, J. Soc. Ind. Appl. Math. 9 (4) (1961) 551–570, URL http://dx.doi.org/10.1137/0109047.
[20] LLC Gurobi Optimization, Gurobi optimizer reference manual, 2023, URL https://www.gurobi.com.
[21] D.S. Johnson, The NP-completeness column: an ongoing guide, J. Algorithms 6 (3) (1985) 434–451, URL http://dx.doi.org/10.1016/0196-6774(85)90012-4.
[22] B. Korte, L. Lovász, R. Schrader, Greedoids, Vol. 4, Springer-Verlag Berlin Heidelberg, 1991, URL http://dx.doi.org/10.1007/978-3-642-58191-5.
[23] E. Lam, V. Mak-Hau, Branch-and-cut-and-price for the cardinality-constrained multi-cycle problem in kidney exchange, Comput. Oper. Res. 115 (2020) 104852, URL http://dx.doi.org/10.1016/j.cor.2019.104852.
[24] A.N. Letchford, G. Reinelt, D.O. Theis, Odd minimum cut sets and b-matchings revisited, SIAM J. Discrete Math. 22 (4) (2008) 1480–1487, URL http://dx.doi.org/10.1137/060664793.
[25] T.L. Magnanti, Wolsey. L.A., Optimal trees, Handb. Oper. Res. Manag. Sci. 7 (1995) 503–615, URL http://dx.doi.org/10.1016/S0927-0507(05)80126-4.
[26] F.K. Miyazawa, P.F. Moura, M.J. Ota, Y. Wakabayashi, Partitioning a graph into balanced connected classes: Formulations, separation and experiments, European J. Oper. Res. 293 (3) (2021) 826–836, URL http://dx.doi.org/10.1016/j.ejor.2020.12.059.
[27] M.W. Padberg, M.R. Rao, Odd minimum cut-sets and b-matchings, Math. Oper. Res. 7 (1) (1982) 67–80, URL http://dx.doi.org/10.1287/moor.7.1.67.

[28] D. Rehfeldt, H. Franz, T. Koch, Optimal connected subgraphs: Integer programming formulations and polyhedra, Networks 80 (3) (2022) 314–332, URL http://dx.doi.org/10.1002/net.22101.
[29] P. Samer, On a class of strong valid inequalities for the connected matching polytope, 2023, URL http://dx.doi.org/10.48550/arXiv.2309.14019 submitted for publication.
[30] L.J. Stockmeyer, V.V. Vazirani, NP-completeness of some generalizations of the maximum matching problem, Inform. Process. Lett. 15 (1) (1982) 14–19, URL http://dx.doi.org/10.1016/0020-0190(82)90077-1.
[31] Y. Wang, A. Buchanan, S. Butenko, On imposing connectivity constraints in integer programs, Math. Program. 166 (2017) 241–271, URL http://dx.doi.org/10.1007/s10107-017-1117-8.