The b-Matching problem in distance-hereditary graphs and beyond

Guillaume Ducoffe, Alexandru Popa

The b-Matching problem in distance-hereditary graphs and beyond [1]

Guillaume Ducoffe[a,b], Alexandru Popa[a,b]

[a] *ICI – National Institute for Research and Development in Informatics, Bucharest, Romania*
[b] *University of Bucharest, Bucharest, Romania*

Abstract

We make progress on the fine-grained complexity of Maximum-Cardinality Matching within graphs of bounded *clique-width*. Quasi linear-time algorithms for this problem have been proposed for the important subclasses of bounded-treewidth graphs (Fomin et al., SODA'17) and graphs of bounded modular-width (Coudert et al., SODA'18). We present such algorithm for bounded *split-width* graphs — a broad generalization of graphs of bounded modular-width, of which an interesting subclass are the distance-hereditary graphs. Specifically, we solve Maximum-Cardinality Matching in $\mathcal{O}((k \log^2 k) \cdot (m+n) \cdot \log n)$-time on graphs with split-width at most k. We stress that the existence of such algorithm was not even known for distance-hereditary graphs until our work. Doing so, we answer an open question of (Coudert et al., SODA'18). Our work brings more insights on the relationships between matchings and *splits*, *a.k.a.*, join operations between two vertex-subsets in different connected components. Furthermore, our analysis can be extended to the more general (unit cost) b-Matching problem. On the way, we introduce new tools for b-Matching and dynamic programming over *split decompositions*, that can be of independent interest.

Keywords: maximum-cardinality matching; b-matching; FPT in P; split decomposition; distance-hereditary graphs.

1. Introduction

The Maximum-Cardinality Matching problem takes as input a graph $G = (V, E)$ and it asks for a subset F of pairwise non incident edges of maximum cardinality. This is a fundamental problem with a wide variety of applications [4, 10, 31, 36]. Hence, the computational complexity of Maximum-Cardinality Matching has been extensively studied in the literature. In [16], Edmonds presented the first polynomial-time algorithm for this problem. It was also the first time that the complexity class P was formalized. Currently, the best-known algorithms for this problem run in $\mathcal{O}(m\sqrt{n})$-time on n-vertex m-edge graphs [34]. Such superlinear running times can be prohibitive for some applications. Intriguingly, Maximum-Cardinality Matching is one of the few remaining fundamental graph problems for which we neither have proved the existence of a quasi linear-time algorithm, nor a superlinear time complexity (conditional) lower-bound. This fact has renewed interest in understanding what kind of graph structure makes this problem difficult. Our present work is at the crossroad of two successful approaches to answer this above question, namely, the quest for improved graph algorithms on special graph classes and the much more recent

[1] Results of this paper were partially presented at the ISAAC'18 conference [14].

program of "FPT in P". We start further motivating these two approaches before we detail our contributions.

1.1. Related work

Algorithmic on special graph classes. One of our initial motivations for this paper was to design a quasi linear-time algorithm for Maximum-Cardinality Matching on *distance-hereditary graphs* [1]. – Recall that a graph G is called distance-hereditary if the distances in any of its connected induced subgraphs are the same as in G. – Distance-hereditary graphs have already been well studied in the literature [1, 13, 25]. In particular, we can solve Diameter in linear-time on this class of graphs [13]. For the latter problem on general graphs, a conditional *quadratic* lower-bound has been proved in [38]. This result suggests that several hard graph problems in P may become easier on distance-hereditary graphs. Our work takes a new step toward better understanding the algorithmic properties of this class of graphs. We stress that there exist linear-time algorithms for computing a maximum matching on several subclasses of distance-hereditary graphs, such as: trees, cographs [42] and (tent,hexahedron)-free distance-hereditary graphs [12]. However, the techniques used for these three above subclasses are quite different from each other. As a byproduct of our main result, we obtain an $\mathcal{O}(m \log n)$-time algorithm for Maximum-Cardinality Matching on distance-hereditary graphs. In doing so, we propose one interesting addition to the list of efficiently solvable special cases for this problem (*e.g.*, see [5, 9, 12, 20, 19, 24, 26, 30, 33, 28, 42, 44, 43]).

Split Decomposition. In order to tackle with Maximum-Cardinality Matching on distance-hereditary graphs, we consider the relationship between this class of graphs and *split decomposition*. A *split* is a join that is also an edge-cut. By using pairwise non crossing splits, termed "strong splits", we can decompose any graph into degenerate and prime subgraphs, that can be organized in a treelike manner. The latter is termed split decomposition [8], and it is our main algorithmic tool for this paper. The *split-width* of a graph is the largest order of a non degenerate subgraph in some canonical split decomposition. In particular, distance-hereditary graphs are exactly the graphs with split-width at most two [37]. Many NP-hard problems can be solved in polynomial time on bounded split-width graphs (*e.g.*, Graph Coloring, see [37]). In particular we have that *clique-width* is upper-bounded by split-width, and so, any FPT algorithm parameterized by clique-width can be transformed into a FPT algorithm parameterized by split-width (the converse is not true). Recently, with Coudert, we designed FPT algorithms for polynomial-time solvable problems when parameterized by split-width [7]. It turns out that many "hard" problems in P such as Diameter can be solved in $\mathcal{O}(k^{\mathcal{O}(1)} \cdot n + m)$-time on graphs with split-width at most k. However, we left this open for Maximum-Cardinality Matching. Indeed, our main contribution in [7] was a Maximum-Cardinality Matching algorithm based on the more restricted *modular decomposition*. Given this previous result, it was conceivable that a Maximum-Cardinality Matching algorithm based on split decomposition could also exist. However, we need to introduce quite different tools than in [7] in order to prove in this work that it is indeed the case.

Fully Polynomial Parameterized Algorithms. Our work with split-width fits in the recent program of "FPT in P". Specifically, given a graph invariant denoted π (in our case, split-width), we address the question whether there exists a Maximum-Cardinality Matching algorithm running in time $\mathcal{O}(k^c \cdot (n + m) \cdot \log^{\mathcal{O}(1)}(n))$, for some constant c, on every graph G such that

$\pi(G) \leq k$ [2]. Note that such an algorithm runs in quasi linear time for any constant k, and that it is faster than the state-of-the art algorithm for Maximum-Cardinality Matching whenever $k = \mathcal{O}(n^{\frac{1}{2c}-\varepsilon})$, for some $\varepsilon > 0$. This kind of FPT algorithms for polynomial problems have attracted recent attention [7, 11, 23, 27, 29, 32] — although some examples can be found earlier in the literature [44, 43]. We stress that Maximum-Cardinality Matching has been proposed in [32] as the "drosophila" of the study of these FPT algorithms in P. We continue advancing in this research direction.

Note that another far-reaching generalization of distance-hereditary graphs are the graphs of bounded *clique-width* [25]. In [7], we initiated the complexity study of Maximum-Cardinality Matching – and other graph problems in P – on bounded clique-width graph classes. The latter research direction was also motivated by the recent $\mathcal{O}(k^2 \cdot n \log n)$-time algorithm for Maximum-Cardinality Matching on graphs of treewidth at most k, see [18, 27]. Turning our attention on denser graph classes of bounded clique-width, we proved in [7] that Maximum-Cardinality Matching can be solved in $\mathcal{O}(k^4 \cdot n + m)$-time on graphs with *modular-width* at most k. We stress that distance-hereditary graphs have *unbounded* treewidth and unbounded modular-width. Furthermore, clique-width is upper-bounded by split-width [37], whereas split-width is upper-bounded by modular-width [7]. As our main contribution in this paper, we present a quasi linear-time algorithm in order to solve some generalization of Maximum-Cardinality Matching on bounded split-width graphs — thereby answering positively to the open question from [7], while improving the state-of-the-art. Our result shows interesting relationships between graph matchings and splits, the latter being an important particular case of the join operation that is used in order to define clique-width. The fine-grained complexity of Maximum-Cardinality Matching parameterized by clique-width, however, remains open.

1.2. Our contributions

We consider a vertex-weighted generalization for Maximum-Cardinality Matching that is known as the unit-cost b-Matching problem [17]. Roughly, every vertex v is assigned some input capacity b_v, and the goal is to compute integer edge-weights $(x_e)_{e \in E}$ so that: for every $v \in V$ the sum of the weights of its incident edges does not exceed b_v, and $\sum_{e \in E} x_e$ is maximized. We prove a simple combinatorial lemma that essentially states that the cardinality of a maximum b-matching in a graph grows as the floor of at most three linear pieces in the capacity b_w of any fixed vertex w. This nice result (apparently never noticed before) holds for any graph. As such, we think that it could provide a nice tool for the further investigations on b-Matching. Then, we derive from our combinatorial lemma a variant of some reduction rule for Maximum-Cardinality Matching that we first introduced in the more restricted case of modular decomposition [7]. Altogether combined, this allows us to reduce the solving of b-Matching on the original graph G to solving b-Matching on *supergraphs* of every its split components. We expect our approach to be useful in other matching and flow problems.

Overall, our main result is that b-Matching can be solved in $\mathcal{O}((k \log^2 k) \cdot (m + n) \cdot \log ||b||_1)$-time on graphs with split-width at most k (Theorem 1). It implies that Maximum-Cardinality Matching can be solved in $\mathcal{O}((k \log^2 k) \cdot (m + n) \cdot \log n)$-time on graphs with split-width at

[2] The polynomial dependency in k is ignored by some authors [32], who accept any functional dependency in the parameter. Since Maximum-Cardinality Matching is already known to be polynomial-time solvable, we find it more natural to consider this restricted setting.

most k. Since distance-hereditary graphs have split-width at most two, we so obtain the first known quasi linear-time algorithms for Maximum-Cardinality Matching and b-Matching on distance-hereditary graphs.

We introduce the required terminology and basic results in Section 2, where we also sketch the main ideas behind our algorithm (Section 2.3). Then, Section 3 is devoted to a combinatorial lemma that is the key technical tool in our subsequent analysis. In Section 4, we present our algorithm for b-Matching on bounded split-width graphs. We conclude in Section 5 with some open questions. Results of this paper were partially presented at the ISAAC'18 conference [14].

2. Preliminaries

We use standard graph terminology from [3]. Graphs in this study are finite, simple (hence without loops or multiple edges), and connected – unless stated otherwise. Furthermore we make the standard assumption that graphs are encoded as adjacency lists. Given a graph $G = (V, E)$ and a vertex $v \in V$, we denote its neighborhood by $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$ and the set of its incident edges by $E_v(G) = \{\{u, v\} \mid u \in N_G(v)\}$. When G is clear from the context we write $N(v)$ and E_v instead of $N_G(v)$ and $E_v(G)$. Similarly, we define the neighborhood of any vertex-subset $S \subseteq V$ as $N_G(S) = \left(\bigcup_{v \in S} N_G(v)\right) \setminus S$.

2.1. Split-width

A complete join is a subset of edges $A \times B$ where A and B are disjoint vertex-subsets (equivalently, the edges of a complete join induce a complete bipartite graph with respective partite sets A and B). Let a *split* in a graph $G = (V, E)$ be a partition $V = U \cup W$ such that there is a complete join between the vertices of $N_G(U)$ and $N_G(W)$. Note that any ordered bipartition (U, W) where $\min\{|U|, |W|\} \leq 1$ is always a split, sometimes called a trivial split. It is desirable to restrict ourselves to non trivial splits, for which we have the additional requirement $\min\{|U|, |W|\} \geq 2$. A *simple decomposition* of G takes as input a non trivial split (U, W), and it outputs two subgraphs $G_U = G[U \cup \{w\}]$ and $G_W = G[W \cup \{u\}]$ where $u, w \notin V$ are fresh new vertices such that $N_{G_U}(w) = N_G(W)$ and $N_{G_W}(u) = N_G(U)$. The vertices u, w are termed *split marker vertices*. A *split decomposition* of G is obtained by applying recursively some sequence of simple decompositions (*e.g.*, see Fig. 1). We name *split components* the subgraphs in a given split decomposition of G.

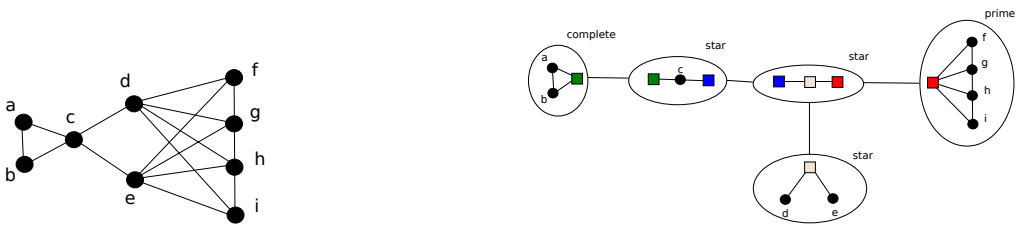

Figure 1: A graph and its split decomposition. Split marker vertices that correspond to a same simple decomposition are identified by two rectangles with the same color.

It is often desirable to apply simple decompositions until all the subgraphs obtained cannot be further decomposed. In the literature there are two cases of "indecomposable" graphs. First, a graph is prime for split decomposition if it only has trivial splits. Throughout the remainder of this paper, a prime subgraph of a graph G refers to any induced subgraph of G that is prime for split decomposition. We can now define the following first type of split decomposition:

- **Minimal split decomposition.** A split-decomposition is *minimal* if all the subgraphs obtained are prime.

The set of prime graphs in any minimal split decomposition is unique up to isomorphism [8]. However, a minimal split decomposition is in general *not* unique. To see that, consider a star with center x_0 and leaves x_1, x_2, x_3. For every $1 \leq i \leq 3$, we can define a split (U_i, W_i) where $U_i = \{x_0, x_i\}$ and $W_i = \{x_1, x_2, x_3\} \setminus \{x_i\}$. The simple decomposition associated to each split leads to the creation of two stars, each having exactly two leaves. However, the outputs of each simple decomposition are pairwise different. It leads us to our second case of "indecomposable" graphs. Specifically, degenerate graphs are such that every bipartition of their vertex-set is a split. They are exactly the complete graphs and the stars [8]. We can now define another type of split decomposition:

- **Canonical split decomposition.** Every graph has a canonical split decomposition where all the subgraphs obtained are either degenerate or prime and the number of subgraphs is minimized. Furthermore, the canonical split decomposition of a given graph can be computed in linear-time [6].

Interestingly, the canonical split decomposition of a graph is unique [8]. The classic algorithms for computing a split decomposition, *e.g.* [6], compute this canonical split decomposition. Nevertheless, a minimal split decomposition can be computed from the canonical split decomposition in linear-time [8]. If the unicity of the decomposition is not required (that is the case for the algorithmic problems considered in this paper), then it is more convenient to work with a minimal split decomposition than with the canonical one. Indeed, doing so, we avoid handling with the particular cases of stars and complete graphs in our algorithms.

For instance, the split decomposition of Fig. 1 is both minimal and canonical for the graph G to its left. Let us replace the edge $\{a, b\}$ by a clique of size four, denoted by K, whose all vertices are adjacent to c. Doing so, we get a new graph G'. In the split decomposition of Fig. 1, there is a unique complete subgraph, whose vertices are a, b and some split marker vertex w. In order to construct the canonical split decomposition of G', it suffices to replace this complete subgraph by a complete subgraph on $K \cup \{w\}$. Since, the canonical split decomposition of G' contains a subgraph with five vertices, it is not minimal. To construct a minimal split decomposition of G', we need to further decompose this subgraph into 3 triangles.

Definition 1. The *split-width* of G, denoted by $sw(G)$, is the minimum $k \geq 2$ such that any prime subgraph in the canonical split decomposition of G has order at most k.

We refer to [37] for some algorithmic applications of split decomposition. In particular, graphs with split-width at most two are exactly the distance-hereditary graphs, *a.k.a* the graphs whose all connected induced subgraphs are distance-preserving [1]. Distance-hereditary graphs contain many interesting subclasses of their own such as *cographs* (*a.k.a.*, P_4-free graphs) and 3-leaf powers. Furthermore, since every degenerate graph has a split decomposition where all the components are either triangles or paths of length three, every component in a minimal split decomposition of G has order at most $\max\{3, sw(G)\}$.

Split decomposition tree. A split decomposition tree of G is a tree T where the nodes are in bijective correspondance with the subgraphs of a given split decomposition of G, and the edges of T are in bijective correspondance with the simple decompositions used for their computation.

More precisely, if the considered split decomposition is reduced to G then T is reduced to a single node; Otherwise, let (U, W) be a split of G and let $G_U = (U \cup \{w\}, E_U),\ G_W = (W \cup \{u\}, E_W)$ be the corresponding subgraphs of G. We construct the split decomposition trees T_U, T_W for G_U and G_W, respectively. Furthermore, the split marker vertices u and w are contained in a unique split component of G_W and G_U, respectively. We obtain T from T_U and T_W by adding an edge between the two nodes that correspond to these subgraphs. The split decomposition tree of the canonical split decomposition, resp. of a minimal split decomposition, can be constructed in linear-time [37].

2.2. Matching problems

A *matching* in a graph is a set of edges with pairwise disjoint end vertices.

Problem 1 (Maximum-Cardinality Matching)**.**

Input: *A graph $G = (V, E)$.*

Output: *A matching of G with maximum cardinality.*

The Maximum-Cardinality Matching problem can be solved in $\mathcal{O}(m\sqrt{n})$-time [34]. We do not use this result directly in our paper. However, we do use in our analysis the notion of *augmenting paths*, that is a cornerstone of most matching algorithms. Namely, let $G = (V, E)$ be a graph and $F \subseteq E$ be a matching of G. A vertex is termed matched if it is incident to an edge of F, and exposed otherwise. An F-augmenting path is a path where the two ends are exposed, all edges $\{v_{2i}, v_{2i+1}\}$ are in F and all edges $\{v_{2j-1}, v_{2j}\}$ are not in F. We can observe that, given an F-augmenting path $P = (v_1, v_2, \ldots, v_{2\ell})$, the matching $E(P)\Delta F$ (obtained by replacing the edges $\{v_{2i}, v_{2i+1}\}$ with the edges $\{v_{2j-1}, v_{2j}\}$) has larger cardinality than F.

Lemma 1 (Berge, [2])**.** *A matching F in $G = (V, E)$ is maximum if and only if there is no F-augmenting path.*

It is folklore that the proof of Berge's lemma also implies the existence of many vertex-disjoint augmenting paths for small matchings. More precisely:

Lemma 2 (Hopcroft-Karp, [26])**.** *Let F_1, F_2 be matchings in $G = (V, E)$. If $|F_1| = r,\ |F_2| = s$ and $s > r$, then there exist at least $s - r$ vertex-disjoint F_1-augmenting paths.*

b-Matching. More generally given a graph $G = (V, E)$, let $b : V \to \mathbb{N}$ assign a nonnegative integer capacity b_v for every vertex $v \in V$. A b-matching is an assignment of nonnegative integer edge-weights $(x_e)_{e \in E}$ such that, for every $v \in V$, we have $\sum_{e \in E_v} x_e \leq b_v$. We define the x-degree of vertex v as $deg_x(v) = \sum_{e \in E_v} x_e$. Furthermore, the cardinality of a b-matching is defined as $||x||_1 = \sum_{e \in E} x_e$. We will consider the following graph problem:

Problem 2 (b-Matching)**.**

Input: *A graph $G = (V, E)$; an assignment function $b : V \to \mathbb{N}$.*

Output: *A b-matching of G with maximum cardinality.*

For technical reasons, we will also use the following variant of b-Matching. Let $c : E \to \mathbb{N}$ assign a cost to every edge. The cost of a given b-matching x is defined as $c \cdot x = \sum_{e \in E} c_e x_e$.

Problem 3 (Maximum-Cost b-Matching)**.**

Input: *A graph $G = (V, E)$; assignment functions $b : V \to \mathbb{N}$ and $c : E \to \mathbb{N}$.*

Output: *A maximum-cardinality b-matching of G where the cost is maximized.*

Lemma 3 ([21, 22])**.** *For every $G = (V, E)$ and $b : V \to \mathbb{N}$, $c : E \to \mathbb{N}$, we can solve* Maximum-Cost b-Matching *in $\mathcal{O}(nm \log^2 n)$-time.*

*In particular, we can solve b-*Matching *in $\mathcal{O}(nm \log^2 n)$-time.*

There is a nonefficient (quasi polynomial) reduction from b-Matching to Maximum-Cardinality Matching that we will use in our analysis (*e.g.*, see [41]). More precisely, let G, b be any instance of b-Matching. The "expanded graph" G_b is obtained from G and b as follows. For every $v \in V$, we add the nonadjacent vertices $v_1, v_2, \ldots, v_{b_v}$ in G_b. Then, for every $\{u, v\} \in E$, we add the edges $\{u_i, v_j\}$ in G_b, for every $1 \leq i \leq b_u$ and for every $1 \leq j \leq b_v$. It is easy to transform any b-matching of G into an ordinary matching of G_b, and vice-versa.

2.3. High-level presentation of the algorithm

In order to discuss the difficulties we had to face on, we start giving an overview of the FPT algorithms that are based on split decomposition.

- We first need to define a vertex-weighted variant of the problem that needs to be solved for every component of the decomposition separately (possibly more than once). This is because there are split marker vertices in every component that substitute the other remaining components; intuitively, the weight of such a vertex encodes a partial solution for the union of split components it has substituted.
- Then, we take advantage of the treelike structure of split decomposition in order to solve the weighted problem, for every split component sequentially, using dynamic programming. Roughly, this part of the algorithm is based on a split decomposition tree. Starting from the leaves of that tree (resp. from the root), we perform a tree traversal. For every split component, we can precompute its vertex-weights from the partial solutions we obtained for its children (resp., for its father) in the split decomposition tree.

Our approach. In our case, a natural vertex-weighted generalization for Maximum-Cardinality Matching is the unit-cost b-Matching problem [17]. Interestingly, b-Matching has already attracted attention on its own, *e.g.*, in auction theory [35, 39]. We observe that independently from this work[3], the authors in [29] proposed a new Maximum-Cardinality Matching algorithm on graphs of bounded modular-width that is also based on a reduction to b-Matching. Unlike this work, the algorithm of [29] cannot be applied to the more general case of bounded split-width graphs. Indeed, the main technical difficulty for the latter graphs – not addressed in [29] – is how to precompute efficiently, for every component of their split decomposition, the specific instances of b-Matching that need to be solved. To see that, consider the bipartition (U, W) that results from the removal of a split. In order to compute the b-Matching instances on side U, we should be able (after processing the other side W) to determine the number of edges of the split that

[3]Our preliminary version of this paper was released on arXiv one day before theirs.

are matched in a final solution. Guessing such number looks computationally challenging. We avoid doing so by storing a partial solution for *every* possible number of split edges that can be matched. However, this simple approach suffers from several limitations. For instance, we need a very compact encoding for partial solutions – otherwise we could not achieve a quasi linear-time complexity. Somehow, we also need to consider the partial solutions for *all* the splits that are incident to the same component all at once.

This is where we use a result from Section 3, namely, that for every fixed vertex w in a graph, the maximum-cardinality of a b-matching is a piecewise-linear function, *with at most three linear pieces*, in the capacity b_w of this vertex. Roughly, in any given split component C_i, we consider all the vertices w substituting a union of other components. The latter vertices are in one-to-one correspondence with the strong splits that are incident to the component. We expand every such vertex w to a module that contains $\mathcal{O}(1)$ vertices for every straight-line section of the corresponding piecewise-linear function. We want to stress that to the best of our knowledge, the combination of dynamic programming over split decomposition with the recursive computation of some piecewise-linear functions is an all new algorithmic technique [4].

3. Changing the capacity of one vertex

We first consider an auxiliary problem on b-matching that can be of independent interest. Let $G = (V, E)$ be a graph, $w \in V$ and $b : V \setminus w \to \mathbb{N}$ be a partial assignment. We denote $\mu(t)$ the maximum cardinality of a b-matching of G provided we set to t the capacity of vertex w. Clearly, μ is nondecreasing in t. Our main result in this section is that the function μ is essentially piecewise linear, with only constantly many pieces (Proposition 1).

We start by introducing some useful lemmata. They are obtained by studying vertex-disjoint augmenting paths in some "expanded graphs" $G_{b,t}$ (cf. Lemmata 1 and 2).

Lemma 4. $\mu(t+1) - \mu(t) \leq 1$.

Proof. Consider the two expanded graphs $G_{b,t}$ and $G_{b,t+1}$ that are obtained after setting the capacity of w to, respectively, t and $t+1$. Let F_{t+1} be a maximum matching of $G_{b,t+1}$. Since $G_{b,t+1}$ can be obtained from $G_{b,t}$ by adding a new vertex w_{t+1}, we can transform F_{t+1} into a (not necessarily maximum) matching of $G_{b,t}$ by removing at most one edge. In particular, $\mu(t) \geq \mu(t+1) - 1$. □

Lemma 5. *If* $\mu(t+2) = \mu(t)$ *then we have* $\mu(t+i) = \mu(t)$ *for every* $i \geq 0$.

Proof. By contradiction, let i_0 be the least integer $i \geq 3$ such that $\mu(t+i) > \mu(t)$. By Lemma 4 we have $\mu(t+i_0) \leq \mu(t+i_0-1) + 1 = \mu(t) + 1$, therefore $\mu(t+i_0) = \mu(t) + 1$. Consider the two expanded graphs $G_{b,t}$ and $G_{b,t+i_0}$ that are obtained after setting the capacity of w to, respectively, t and $t+i_0$. Let F_t be any maximum matching of $G_{b,t}$. By the hypothesis, $|F_t| = \mu(t)$. Furthermore, since $G_{b,t+i_0}$ can be obtained from $G_{b,t}$ by adding the new vertices $w_{t+1}, \ldots, w_{t+i_0}$ (that are false twins of $w_1, w_2, \ldots, w_t$), F_t is also a matching of $G_{b,t+i_0}$. However F_t is not maximum in $G_{b,t+i_0}$ since we have $\mu(t+i_0) = \mu(t) + 1$. By Berge's lemma (Lemma 1) there exists an F_t-augmenting path P in $G_{b,t+i_0}$. Furthermore, since F_t is maximum in $G_{b,t}$, P must contain a vertex amongst

[4] Earlier examples of dynamic programming which recursively computes a piece-wise linear function can be found in the literature, *e.g.*, see [40].

$w_{t+1}, \ldots, w_{t+i_0}$. Note that the latter vertices are all exposed (since they are not in $G_{b,t}$), and so, an F_t-augmenting path can only contain at most two of them. Furthermore, since $w_{t+1}, \ldots, w_{t+i_0}$ are pairwise false twins, we can assume w.l.o.g. that P has w_{t+1} as an end. In the same way, in case P has its two ends amongst $w_{t+1}, \ldots, w_{t+i_0}$ then we can assume w.l.o.g. that the two ends of P are exactly w_{t+1}, w_{t+2}. It implies either $\mu(t) < \mu(t+1) \leq \mu(t+2)$ or $\mu(t) = \mu(t+1) < \mu(t+2)$. In both cases, $\mu(t+2) > \mu(t)$, that is a contradiction. □

Lemma 6. *If $\mu(t+1) = \mu(t)$ then we have $\mu(t+3) = \mu(t+2)$.*

Proof. If $\mu(t+2) = \mu(t)$ then the result follows from Lemma 5 directly. Thus, we assume from now on that $\mu(t+2) > \mu(t)$. By Lemma 4 we have $\mu(t+2) \leq \mu(t+1) + 1 = \mu(t) + 1$, therefore $\mu(t+2) = \mu(t) + 1$. Suppose by contradiction $\mu(t+3) > \mu(t+2)$. Again by Lemma 4 we get $\mu(t+3) = \mu(t+2) + 1$, therefore $\mu(t+3) = \mu(t) + 2$. Consider the two expanded graphs $G_{b,t}$ and $G_{b,t+3}$ that are obtained after setting the capacity of w to, respectively, t and $t+3$. Let F_t be any maximum matching of $G_{b,t}$. By the hypothesis, $|F_t| = \mu(t)$. Furthermore, since $G_{b,t+3}$ can be obtained from $G_{b,t}$ by adding the three new vertices $w_{t+1}, w_{t+2}, w_{t+3}$ (that are false twins of $w_1, w_2, \ldots, w_t$), F_t is also a matching of $G_{b,t+3}$. However, F_t is not maximum in $G_{b,t+3}$ since we have $\mu(t+3) = \mu(t) + 2$. By Hopcroft-Karp lemma (Lemma 2) there exist 2 F_t-augmenting paths in $G_{b,t+3}$ that are vertex-disjoint. Furthermore, since F_t is maximum in $G_{b,t}$, every such path must contain a vertex amongst $w_{t+1}, w_{t+2}, w_{t+3}$. Note that the latter vertices are all exposed, and so, an F_t-augmenting path can only contain at most two of them. In particular, there is at least one of these two F_t-augmenting paths, denoted by P, that only contains a single vertex amongst $w_{t+1}, w_{t+2}, w_{t+3}$. W.l.o.g., since $w_{t+1}, w_{t+2}, w_{t+3}$ are pairwise false twins, we can assume that w_{t+1} is an end of P. However, it implies that P is also an F_t-augmenting path in $G_{b,t+1}$, and so, that $\mu(t+1) > \mu(t)$, that is a contradiction. □

We are now ready to prove the main result in this section:

Proposition 1. *There exist integers c_1, c_2 such that:*

$$\mu(t) = \begin{cases} \mu(0) + t \text{ if } t \leq c_1 \\ \mu(c_1) + \left\lfloor \frac{t-c_1}{2} \right\rfloor = \mu(0) + c_1 + \left\lfloor \frac{t-c_1}{2} \right\rfloor \text{ if } c_1 < t \leq c_1 + 2c_2 \\ \mu(c_1 + 2c_2) = \mu(0) + c_1 + c_2 \text{ otherwise.} \end{cases}$$

Furthermore, the triple $(\mu(0), c_1, c_2)$ can be computed in $\mathcal{O}(nm \log^2 n \log ||b||_1)$-time.

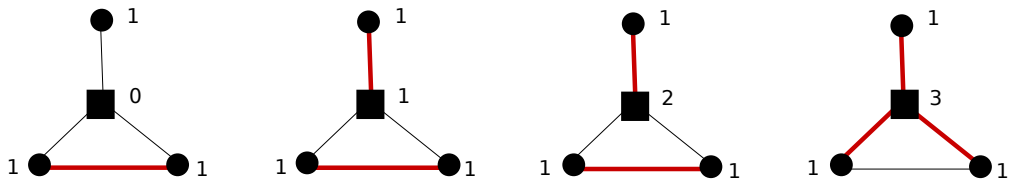

Figure 2: An example with $(\mu(0), c_1, c_2) = (1, 1, 1)$. Vertices are labeled with their capacity. Thin and bold edges have respective weights 0 and 1.

Proof. Let c_1 be the maximum integer t such that $\mu(t) = \mu(0) + t$. This value is well-defined since μ must stay constant whenever $t \geq \sum_{v \in N_G(w)} b_v$ (saturation of all the neighbors). Furthermore,

by Lemma 4 we have $\mu(t) = \mu(0) + t$ for every $0 \leq t \leq c_1$. Then, let $t_{\max}$ be the least integer t such that, for every $i \geq 0$ we have $\mu(t_{\max} + i) = \mu(t_{\max})$. Again, this value is well-defined since we have the trivial upper-bound $t_{\max} \leq \sum_{v \in N_G(w)} b_v$. Furthermore, since μ is strictly increasing between 0 and c_1, $t_{\max} \geq c_1$. Let $c'_2 = t_{\max} - c_1$. We claim that $c'_2 = 2c_2$ is even. For that, we need to observe that $\mu(c_1) = \mu(c_1 + 1)$ by maximality of c_1. Using Lemma 6, we prove by induction $\mu(c_1 + 2i) = \mu(c_1 + 2i + 1)$ for every $i \geq 0$. The latter proves, as claimed, $c'_2 = 2c_2$ is even by minimality of c'_2. Moreover, for every $0 \leq i < c_2$ we have by Lemma 5 $\mu(c_1 + 2i) < \mu(c_1 + 2(i + 1))$ (since otherwise $t_{\max} \leq c_1 + 2i$). By Lemma 6 we have $\mu(c_1 + 2i) = \mu(c_1 + 2i + 1)$. Finally, by Lemma 4 we get $\mu(c_1 + 2(i + 1)) \leq \mu(c_1 + 2i + 1) + 1 = \mu(c_1 + 2i) + 1$, therefore $\mu(c_1 + 2(i + 1)) = \mu(c_1 + 2i) + 1$. Altogether combined, it implies that $\mu(c_1 + 2i) = \mu(c_1 + 2i + 1) = \mu(c_1) + i$ for every $0 \leq i \leq c_2$, that proves the first part of our result.

We can compute $\mu(0)$ with any b-Matching algorithm after we set the capacity of w to 0. The value of c_1 can be computed within $\mathcal{O}(\log c_1)$ calls to a b-Matching algorithm, as follows. Starting from $c'_1 = 1$, we multiply the current value of c'_1 by 2 until we reach a value $c'_1 > c_1$ such that $\mu(c'_1) < \mu(0) + c'_1$. Then, we perform a binary search between 0 and c'_1 in order to find the largest value c_1 such that $\mu(c_1) = \mu(0) + c_1$. Once c_1 is known, we can use a similar approach in order to compute c_2. Overall, since $c_1 + 2c_2 = t_{\max} \leq \sum_{v \in N_G(w)} b_v = \mathcal{O}(||b||_1)$, we are left with $\mathcal{O}(\log ||b||_1)$ calls to any b-Matching algorithm. Therefore, by Lemma 3, we can compute the triple $(\mu(0), c_1, c_2)$ in $\mathcal{O}(nm \log^2 n \log ||b||_1)$-time. □

4. The algorithm

We present in this section a quasi linear-time algorithm for computing a maximum-cardinality b-matching on any bounded split-width graph (Theorem 1). Given a graph G, our algorithm takes as input the split decomposition tree T of any minimal split decomposition of G. We root T in an arbitrary component C_1. Then, starting from the leaves, we compute by dynamic programming on T the *cardinality* of an optimal solution. This first part of the algorithm is involved, and it uses the results of Section 3. It is based on a new reduction rule that we introduce in Definition 2. Finally, starting from the root component C_1, we compute a maximum-cardinality b-matching of G, b by top-to-bottom dynamic programming on T. This second part of the algorithm is simpler than the first one, but we need to carefully upper-bound its time complexity. In particular, we also need to ensure that some additional property holds for the b-matchings we compute at every component.

4.1. Reduction rule

Recall that an edge between a rooted subtree and its parent in T corresponds to a split (U, W) of G. After we processed the side U (corresponding to this subtree) we account for all the partial solutions found for G_U by transforming the split marker vertex u into a *module* [5], as follows:

Definition 2. For any instance $G = (V, E), b$ and any split (U, W) of G let $C = N_G(W) \subseteq U$, $D = N_G(U) \subseteq W$. Let $G_U = (U \cup \{w\}, E_U)$, $G_W = (W \cup \{u\}, E_W)$ be the corresponding subgraphs of G. We define the pairs G_U, b^U and H_W, b^W as follows:

- For every $v \in U$ we set $b^U_v = b_v$; the capacity of the split marker vertex w is left unspecified. Let $(\mu^U(0), c^U_1, c^U_2)$ be as defined in Proposition 1 w.r.t. G_U, b^U and w.

[5] Recall that M is a module if for every $x, y \in M$ we have $N(x) \setminus M = N(y) \setminus M$.

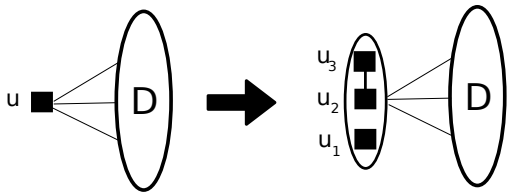

Figure 3: The reduction of Definition 2.

- The *auxiliary graph* H_W is obtained from G_W by replacing the split marker vertex u by a module $M_u = \{u_1, u_2, u_3\}$, $N_{H_W}(M_u) = N_{G_W}(u) = D$; we also add an edge between u_2, u_3. For every $v \in W$ we set $b^W_v = b_v$; we set $b^W_{u_1} = c^U_1$, $b^W_{u_2} = b^W_{u_3} = c^U_2$.

See Fig. 3 for an illustration. We will show throughout this section that our gadget somewhat encodes all the partial solutions for side U. Formally, the following relationship holds between solutions for G, b and solutions for H_W, b^W:

Proposition 2. *Given a graph $G = (V, E)$ and a capacity function b, let (U, W) be a split of G and let H_W, b^W be as in Definition 2. If x and x^W are maximum-cardinality b-matchings for the pairs G, b and H_W, b^W, respectively, then we have:*

$$||x||_1 = ||x^W||_1 + \mu^U(0) - c^U_2$$

In what follows, we prove the first direction of Proposition 2 using classical flow techniques. We postpone the proof of the other direction since, for that one, we need to prove intermediate lemmata that will be also used in the proof of Theorem 1.

Lemma 7. *Let x be a b-matching for G, b. There exists a b-matching x^W for H_W, b^W such that $||x^W||_1 \geq ||x||_1 + c^U_2 - \mu^U(0)$.*

Proof. See Fig. 4 for an illustration. Let us define $b^U_w = \sum_{e \in C \times D} x_e$. As an intermediate step, we can construct a b-matching x^U of the pair G_U, b^U as follows. First we keep all the edge-weights $x_e = x^U_e$, for every $e \in E(U)$. Then, for every $v \in C$ we set $x^U_{\{v,w\}} = \sum_{v' \in D} x_{\{v,v'\}}$. We deduce from this transformation that:

$$||x^U||_1 = \sum_{e \in E(U) \cup (C \times D)} x_e \leq \mu^U(b^U_w).$$

In particular, let y^U be a b-matching of G_U, b^U of optimal cardinality $\mu^U(b^U_w)$ and such that $d = deg_{y^U}(w) = \sum_{e \in E_w(G_U)} y^U_e$ is minimized. By Proposition 1, we have $d \leq c^U_1 + 2c^U_2$. We obtain a b-matching x^W for the pair H_W, b^W as follows:

- We keep all the edge-weights $x_e = x^W_e$, for every $e \in E(W)$. Doing so, we get an initial b-matching of cardinality $||x||_1 - ||x^U||_1 \geq ||x||_1 - \mu^U(b^U_w)$;
- Then, in order to define x^W_e, for every edge e that is incident to u_1, we make a simple reduction to a flow problem. Namely, consider a star with leaves D and central node the split marker vertex u. The star is node-capacitated, with the capacity being set to: $\min\{d, c^U_1\}$ for u; and $\sum_{v \in C} x_{\{v,v'\}}$ for every $v' \in D$. By construction, the capacities on side D sum to $b^U_w \geq d$. So,

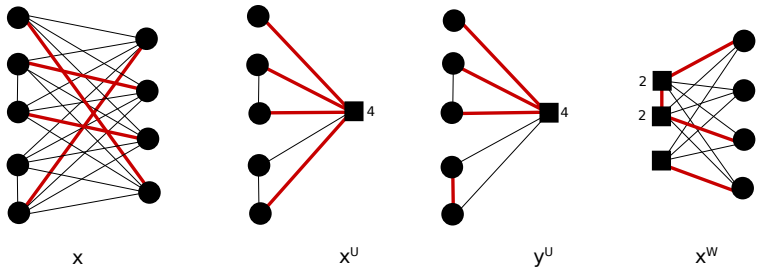

Figure 4: The construction of x^W. Vertices with capacity greater than 1 are labeled with their capacity. Thin and bold edges have respective weights 0 and 1.

we can send exactly $\min\{d, c_1^U\}$ units of flow from D to u. Furthermore, we define $x^W_{\{u_1,v'\}}$, for every $v' \in D$, as the value of the flow passing through the arc (v', u). Doing so, we get a partial solution of cardinality $\geq ||x||_1 - \mu^U(b_w^U) + \min\{d, c_1^U\}$.

- We end up defining the weights of edges incident to u_2, u_3. There are two cases:
  - **Case $d < c_1^U$.** By Proposition 1, $b_w^U = d$ and $\mu^U(b_w^U) - d = \mu^U(0)$. We complete the b-matching by setting $x^W_{\{u_2,u_3\}} = c_2^U$.
  - **Case $d \geq c_1^U$.** By Proposition 1 and the minimality of d, we have that $d - c_1^U = 2d'$ is even, $d' \leq c_2^U$ and $\mu^U(b_w^U) - c_1^U = \mu^U(0) + d'$. We make two other reductions to a flow problem, on the same star as before but with different node capacities. More precisely, we set the capacity of u to d', while for every $v' \in D$, we decrease its original capacity by $x^W_{\{u_1,v'\}}$. By construction, the capacities on side D now sum to $b_w^U - c_1^U \geq d - c_1^U \geq 2d'$. So, we can send exactly d' units of flow from D to u. Furthermore, we define $x^W_{\{u_2,v'\}}$, for every $v' \in D$, as the value of the flow passing through the arc (v', u).
    Then, we again decrease the node capacity for every $v' \in D$, by exactly $x^W_{\{u_2,v'\}}$. We send d' more units of flow from D to u. For every $v' \in D$, we define $x^W_{\{u_3,v'\}}$ as the additional amount of flow being sent through the arc (v', u). Note that, doing so, we get that $\sum_{v' \in D} x^W_{\{u_2,v'\}} = \sum_{v' \in D} x^W_{\{u_3,v'\}} = d'$. Finally, we set $x^W_{\{u_2,u_3\}} = c_2^U - d'$. In total, the cardinality of the b-matching has so increased by $2d' + (c_2^U - d') = c_2^U + d'$.

In both cases, the resulting b-matching has cardinality at least $||x||_1 + c_2^U - \mu^U(0)$.

□

The following Sections 4.2 and 4.3 detail the intermediate results that we will use in order to prove the other direction of Proposition 2 (as well as Theorem 1).

4.2. b-matchings with additional properties

We consider an intermediate modification problem on the b-matchings of some "auxiliary graphs" that we define next. – The necessary notations and terminology introduced here for understanding this intermediate problem are repeated in the proof of Theorem 1. – First, let us fix a given split decomposition of G, and let T be the associated split decomposition tree. We root T arbitrarily. Let C_i be an arbitrary split component. We have that C_i is obtained from a sequence of simple decompositions. Specifically:

- Let $C_{p(i)}$ be the parent node of C_i (if it exists). The edge $\{C_i, C_{p(i)}\} \in E(T)$ corresponds to some split (U_i, W_i) of G, where $V(C_i) \cap V \subseteq U_i$. Recall that after we applied the simple decomposition associated to this split (see Sec. 2), we created the two split marker vertices u_i, w_i. In particular $w_i \in V(C_i)$.
- In the same way, if $\{C_i, C_{i_t}\}$ is an edge of T between C_i and a child node C_{i_t}, then it corresponds to some split (U_{i_t}, W_{i_t}) of G, where $V(C_i) \cap V \subseteq W_{i_t}$. After we applied the simple decomposition associated to this split (see Sec. 2), we created the two split marker vertices u_{i_t}, w_{i_t}. In particular $u_{i_t} \in V(C_i)$.

In order to create an auxiliary graph, we consider a split component C_i and the subsequence of all the simple decompositions that are corresponding to the edges between C_i and its ℓ children C_{i_t} in T (*i.e.*, we only exclude the edge between C_i and its parent node in T if it exists). We apply the reduction rule of Definition 2 to each simple decomposition of this subsequence. Doing so, we obtain a pair H_i, b^{H_i} with H_i being a supergraph of C_i obtained by replacing the split marker vertices $u_{i_t},\ 1 \leq t \leq \ell$, by the modules $M_{i_t} = \{u^1_{i_t}, u^2_{i_t}, u^3_{i_t}\}$. By construction the capacity of vertex $u^1_{i_t}$ equals $c^{i_t}_1$; furthermore, $u^2_{i_t}, u^3_{i_t}$ are adjacent and they have the same capacity $c^{i_t}_2$.

Let $w_i \in V(C_i)$ be the split marker vertex associated to the edge between C_i and its parent node in T (if it exists). We observe that our above construction does not say anything about the capacity of w_i in H_i. Actually, several different capacities need to be assigned to w_i in our final algorithm (*i.e.*, see the proof of Theorem 1). However, this has no incidence on the discussion below, and so, for the remainder of this section, we may assume the capacity of w_i to be arbitrary.

We seek for a maximum-cardinality b-matching x^i for the pair H_i, b^{H_i} such that the following properties hold for every $1 \leq t \leq \ell$:

- (**symmetry**) $deg_{x^i}(u^2_{i_t}) = deg_{x^i}(u^3_{i_t})$.
- (**saturation**) if $deg_{x^i}(u^1_{i_t}) < c^{i_t}_1$ then, $deg_{x^i}(u^2_{i_t}) = x^i_{\{u^2_{i_t}, u^3_{i_t}\}}$.

We prove next that for every fixed t, any x^i can be processed in $\mathcal{O}(|E_{u_{i_t}}(C_i)|)$-time so that both the saturation property and the symmetry property hold for M_{i_t}. However, ensuring that these two above properties hold *simultaneously* for every t happens to be trickier. We manage to do so by reducing to Maximum-Cost b-Matching (*i.e.*, internal edges in the modules are assigned a larger cost than the other edges).

Lemma 8. *In $\mathcal{O}(|V(H_i)| \cdot |E(H_i)| \cdot \log^2 |V(H_i)|)$-time, we can compute a maximum-cardinality b-matching x^i for the pair H_i, b^{H_i} such that both the saturation property and the symmetry property hold for every $M_{i_t},\ 1 \leq t \leq \ell$.*

Proof. We start presenting an iterative solution that works on a standard (unit-cost) b-matching. We show why this solution may not work when there are $\ell > 1$ modules. Then, we explain how to fix this solution by assigning edge-costs.

Let x^i be some initial maximum-cardinality b-matching (to be defined later). While there exists a t such that the saturation property or the symmetry property does not hold for M_{i_t}, we keep repeating the following rules until none of them can be applied:

- **Rule 1**. Suppose $deg_{x^i}(u^1_{i_t}) < c^{i_t}_1$ and there exists $v' \in N_{H_i}(M_{i_t})$ such that $x^i_{\{u^2_{i_t}, v'\}} > 0$ (resp., $x^i_{\{u^3_{i_t}, v'\}} > 0$). Then, we increase $x^i_{\{u^1_{i_t}, v'\}}$ as much as we can, that is by exactly

$\min\{c_1^{i_t} - deg_{x^i}(u_{i_t}^1), x^i_{\{u_{i_t}^2, v'\}}\}$ (resp., $\min\{c_1^{i_t} - deg_{x^i}(u_{i_t}^1), x^i_{\{u_{i_t}^3, v'\}}\}$), and we decrease $x^i_{\{u_{i_t}^2, v'\}}$ (resp., $x^i_{\{u_{i_t}^3, v'\}}$) by exactly the same amount. By repeating this step until it is no more possible to do so, we ensure that the saturation property holds for M_{i_t}.

- **Rule 2**. Suppose $deg_{x^i}(u_{i_t}^2) > deg_{x^i}(u_{i_t}^3)+1$ (the case $deg_{x^i}(u_{i_t}^3) > deg_{x^i}(u_{i_t}^2)+1$ is symmetrical to this one). Let $v' \in N_{H_i}(M_{i_t})$ such that $x^i_{\{u_{i_t}^2, v'\}} > x^i_{\{u_{i_t}^3, v'\}}$. We increase $x^i_{\{u_{i_t}^3, v'\}}$ as much as we can, that is by $\min\left\{\left\lfloor \frac{deg_{x^i}(u_{i_t}^2) - deg_{x^i}(u_{i_t}^3)}{2} \right\rfloor, x^i_{\{u_{i_t}^2, v'\}}\right\}$, while we decrease $x^i_{\{u_{i_t}^2, v'\}}$ by exactly the same amount. By repeating this step until it is no more possible to do so, we ensure that $|deg_{x^i}(u_{i_t}^2) - deg_{x^i}(u_{i_t}^3)| \leq 1$.
- **Rule 3**. Suppose $deg_{x^i}(u_{i_t}^2) = deg_{x^i}(u_{i_t}^3)+1$ (the case $deg_{x^i}(u_{i_t}^3) = deg_{x^i}(u_{i_t}^2)+1$ is symmetrical to this one). Let $v' \in N_{H_i}(M_{i_t})$ such that $x^i_{\{u_{i_t}^2, v'\}} > x^i_{\{u_{i_t}^3, v'\}}$. We decrease $x^i_{\{u_{i_t}^2, v'\}}$ by one unit; similarly, we increase $x^i_{\{u_{i_t}^2, u_{i_t}^3\}}$ by one unit. Doing so, we ensure that the symmetry property holds for M_{i_t}.

Overall, we only need to scan $\mathcal{O}(1)$ times the set $N_{H_i}(M_{i_t})$, and so, we can ensure that both the saturation property and the symmetry property hold for M_{i_t} in $\mathcal{O}(|N_{H_i}(M_{i_t})|)$-time. However, doing so, we may break the saturation property or the symmetry property for some other $t' \neq t$ (*e.g.*, see Fig. 5). Therefore, if we start from an arbitrary x^i, the above procedure may take quasi polynomial time in order to converge.

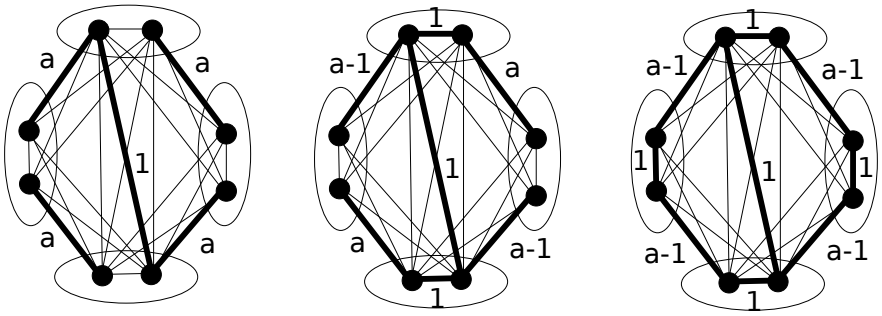

Figure 5: An example where the naive processing stage requires $\mathcal{O}(||b||_1)$-time.

In order to overcome this above issue, we assign costs on the edges. All the edges of H_i have unit cost, except the edges $\{u_{i_t}^2, u_{i_t}^3\}$, for every $1 \leq t \leq \ell$, to which we assign cost 2. We compute a maximum-cardinality b-matching x^i for the pair H_i, b^{H_i} that is of *maximum cost*. By Lemma 3, this can be done in time $\mathcal{O}(|V(H_i)||E(H_i)| \log^2 |V(H_i)|)$. Then, we apply the same procedure on x^i as described above. We claim that there is at most one loop for every t. Indeed, let $1 \leq t \leq \ell$ be arbitrary. We observe that Rules 1 and 2 cannot modify the weight of an edge *inside* a module M_{i_j}. As a result, Rules 1 and 2 do not change the cost of the b-matching. Furthermore, for every $v' \in N_{H_i}(M_{i_t})$, Rules 1 and 2 do not change the x^i-degree of v'. Hence, we can only break the saturation property or the symmetry property for some other $t' \neq t$ by applying Rule 3. However, Rule 3 increases the cost of x^i, and so, since x^i is supposed to be of maximum cost, this rule will never be applied. Therefore, the claim is proved. Overall, it implies that the postprocessing of x^i takes time $\mathcal{O}(\sum_{t=1}^{\ell} |N_{H_i}(M_{i_t})|)$, that is in $\mathcal{O}(|E(H_i)|)$. □

4.3. Merging the partial solutions together

Finally, before we can describe our main algorithm (Theorem 1) we need to consider the intermediate problem of merging two partial solutions. Let (U, W) be a split of G and let $G_U = (U \cup \{w\}, E_U)$, $G_W = (W \cup \{u\}, E_W)$ be the corresponding subgraphs of G. Consider some partial solutions x^U and x^W obtained, respectively, for the pairs G_U, b^U and G_W, b^W (for some b^U, b^W to be defined later). Assuming an appropriate data-structure for b-matchings (Lemma 9), this merging stage of x^U with x^W can be solved with a greedy algorithm (Lemma 10).

First we introduce the following data structure for storing a b-matching. To make things easier in our presentation, we shall assume (*only* for Lemmata 9 and 10 below) the split marker vertices u and w, for any split (U, W), to be vertices of the original graph G. Specifically, we assume that we have $u \in N_G(W)$ and $w \in N_G(U)$. Note that in this situation, $E_U \cup E_W \subseteq E(G)$, and in fact $E_U \cap E_W$ is reduced to a single edge $e = \{u, w\}$.

Lemma 9. *For every $G = (V, E)$ there exists a data structure that can be initialized in $\mathcal{O}(m)$-time, and such that the following properties hold:*

- *(***Access***) An edge-weight assignment $(x_e)_{e \in E}$ is stored. Initially, all the edge-weights are set to 0. For every $e \in E$, we can read and modify x_e in constant-time.*
- *(***Split***) Furthermore, let (U, W) be a split of G and let $G_U = (U \cup \{w\}, E_U)$, $G_W = (W \cup \{u\}, E_W)$ be the corresponding subgraphs of G. In $\mathcal{O}(1)$-time, we can modify the data structure so that it can store separate edge-weight assignments for G_U and G_W (initially set to $(x_e)_{e \in E_U}$ and $(x_e)_{e \in E_W}$).*
- *(***Merge***) Conversely, let (U, W) be a split of G and let $G_U = (U \cup \{w\}, E_U)$, $G_W = (W \cup \{u\}, E_W)$ be the corresponding subgraphs of G. Suppose that two separate assignments $(x^U_e)_{e \in E_U}$ and $(x^W_e)_{e \in E_W}$ are stored in the data structure. In $\mathcal{O}(1)$-time, we can modify the data structure so that it stores the following edge-weight assignment for G:*

$$x_e = \begin{cases} x^U_e \text{ if } e \in E_U \setminus E_W \\ x^W_e \text{ if } e \in E_W \setminus E_U \\ \max\{x^U_e, x^W_e\} \text{ if } e \in E_U \cap E_W \\ \text{undefined otherwise.} \end{cases}$$

Proof. Every edge $e \in E$ has a pointer to its corresponding weight x_e (initially set to 0). Now, consider any split (U, W) of G and let $G_U = (U \cup \{w\}, E_U)$, $G_W = (W \cup \{u\}, E_W)$ be the corresponding subgraphs of G. Observe that E_U, E_W intersect in exactly one edge $e_{U,W} = \{u, w\}$. So, in order to perform a split of the data structure, it suffices to split the pointer of $e_{U,W}$ in two new pointers, that are initially pointing to two distinct copies of the value x_e. Note that for every pointer newly introduced, we need to keep track of the corresponding split (U, W) and of the corresponding side (*i.e.*, U or W). Conversely, in order to merge the respective data structures obtained for G_U and G_W, it suffices to extract the values $x^U_{e_{U,W}}$ and $x^W_{e_{U,W}}$ (on which the two new pointers introduced after the split are pointing to) and to set the original value $x_{e_{U,W}}$ to $\max\{x^U_{e_{U,W}}, x^W_{e_{U,W}}\}$. □

Lemma 10. *Suppose that b^U (resp., b^W) satisfies $b^U_v \leq b_v$ for every $v \in U$ (resp., $b^W_v \leq b_v$ for every $v \in W$). Let x^U, x^W be b-matchings for, respectively, the pairs G_U, b^U and G_W, b^W such that we*

have $deg_{x^U}(w) = deg_{x^W}(u) = d$. Furthermore, for any graph H let $\varphi(H) = |E(H)| + 4 \cdot (sc(H) - 1)$, with $sc(H)$ being the number of split components in any minimal split decomposition of H [6].

Then, in at most $K \cdot (\varphi(G) - \varphi(G_U) - \varphi(G_W))$-time, for some universal constant K, we can obtain a valid b-matching x for the pair G, b such that $||x||_1 = ||x^U||_1 + ||x^W||_1 - d$.

Proof. There are two cases. First, suppose $C = N_G(W) = \{u\}$ (the case $D = N_G(U) = \{w\}$ is symmetrical to this one). The split marker vertex w is pendant in G_U, with its unique neighbor being u (so, in particular, $x^U_{\{u,w\}} = d$). In order to compute x, we set $x^U_{\{u,w\}}$ to 0 and then we merge x^U, x^W. By Lemma 9 this takes constant-time. Furthermore, since $|E| - |E_U| - |E_W| = |C||D| - |C| - |D| = -1$, and $sc(G) = sc(G_U) + sc(G_W)$, we get $\varphi(G) - \varphi(G_U) - \varphi(G_W) = 3 > 0$.

Therefore, from now on we assume that $|C| \geq 2$ and $|D| \geq 2$. For every $v \in C$ we assign a capacity $c_v = x^U_{\{v,w\}}$ and then we set $x^U_{\{v,w\}}$ to 0. In the same way, for every $v' \in D$ we assign a capacity $c_{v'} = x^W_{\{v',u\}}$ and then we set $x^W_{\{v',u\}}$ to 0. It takes $\mathcal{O}(|C| + |D|)$-time. Overall, $\sum_{v \in C} c_v = \sum_{v' \in D} c_{v'} = d$. Then, let us merge x^U, x^W in order to initialize x (note that by construction, $x_{\{u,w\}} = 0$, and $||x||_1 = ||x^U||_1 + ||x^W||_1 - 2d$). By Lemma 9 this takes constant-time. While there exist a $v \in C$ and a $v' \in D$ such that $c_v > 0,\ c_{v'} > 0$ we pick one such pair v, v' and we set: $x_{\{v,v'\}} = \min\{c_v, c_{v'}\};\ c_v = c_v - x_{\{v,v'\}};\ c_{v'} = c_{v'} - x_{\{v,v'\}}$. Since for every loop, the capacity of at least one vertex drops to 0, it takes total time $\mathcal{O}(|C| + |D|)$. Furthermore, since $|C| \geq 2$ and $|D| \geq 2$ we have $|E| - |E_U| - |E_W| = |C||D| - (|C| + |D|) \geq 2(|C| + |D|) - 4 - (|C| + |D|) \geq |C| + |D| - 4$. As a result, $\varphi(G) - \varphi(G_U) - \varphi(G_W) \geq |C| + |D| - 4 + 4 \geq \Omega(|C| + |D|)$. □

Overall, since there are at most $n - 2$ components in any minimal split decomposition of G [37], the merging stages take total time $\mathcal{O}(\varphi(G)) = \mathcal{O}(n + m)$.

4.4. Main result

The following algorithmic proof of Proposition 2 is the cornerstone of our main result.

Proof of Proposition 2. We have $||x^W||_1 \geq ||x||_1 - \mu^U(0) + c^U_2$ by Lemma 7. In order to prove the converse inequality, we can assume w.l.o.g. that x^W satisfies both the saturation property and the symmetry property w.r.t. the module M_u (otherwise, by Lemma 8, we can process x^W so that it is the case). We partition $||x^W||_1$ as follows: $\mu^W = \sum_{e \in E(W)} x^W_e$, $c'_1 = deg_{x^W}(u_1) \leq c^U_1$ and $c'_2 = deg_{x^W}(u_2) - x^W_{\{u_2,u_3\}} = deg_{x^W}(u_3) - x^W_{\{u_2,u_3\}} \leq c^U_2$. Since we assume that x^W satisfies both the saturation property and the symmetry property w.r.t. M_u, we have $c'_2 > 0$ only if $c'_1 = c^U_1$. Furthermore, we observe that u_2 and u_3 must be saturated (otherwise, we could increase the cardinality of the b-matching by setting $x^W_{\{u_2,u_3\}} = c^U_2 - c'_2$). Therefore, we get:

$$||x^W||_1 = \mu^W + c'_1 + 2c'_2 + (c^U_2 - c'_2) = \mu^W + c'_1 + c'_2 + c^U_2.$$

We define $b^W_u = b^U_w = c'_1 + 2c'_2$. Then, we proceed as follows (see Fig. 6 for an illustration).

- We transform x^W into a b-matching for the pair G_W, b^W by setting $x^W_{\{u,v'\}} = x^W_{\{u_1,v'\}} + x^W_{\{u_2,v'\}} + x^W_{\{u_3,v'\}}$ for every $v' \in N_{G_W}(u) = D$. Note that we have $deg_{x^W}(u) = b^W_u = c'_1 + 2c'_2$. Furthermore, the cardinality of the b-matching has decreased by $x^W_{\{u_2,u_3\}} = c^U_2 - c'_2$.

[6] We recall that the set of prime graphs in any minimal split decomposition is unique up to isomorphism [37].

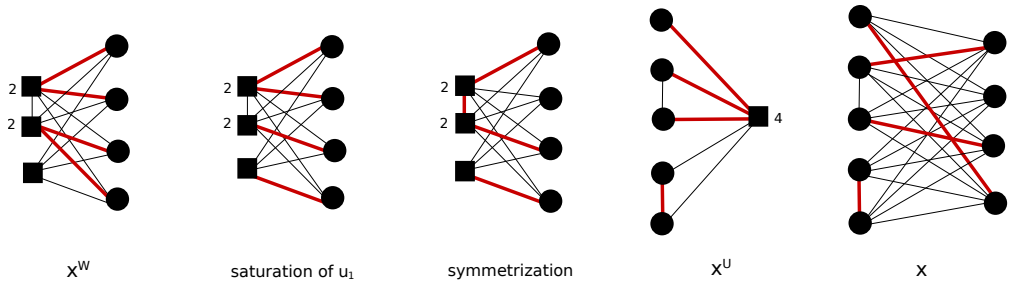

Figure 6: The construction of x'. Vertices with capacity greater than 1 are labeled with their capacity. Thin and bold edges have respective weights 0 and 1.

- Let x^U be a b-matching for the pair G_U, b^U of maximum cardinality $\mu^U(c'_1 + 2c'_2)$. Since $c'_1 \leq c_1^U$, $c'_2 > 0$ only if $c'_1 = c_1^U$, and $c'_2 \leq c_2^U$, the following can be deduced from Proposition 1: $||x^U||_1 = \mu^U(c'_1 + 2c'_2) = \mu^U(0) + c'_1 + c'_2$; and the split marker vertex w is saturated in x^U, *i.e.*, $deg_{x^U}(w) = b_w^U = c'_1 + 2c'_2$.

Since we have $deg_{x^W}(u) = deg_{x^U}(w) = c'_1 + 2c'_2$, we can define a b-matching x' for the pair G, b by applying Lemma 10. Doing so, we get $||x||_1 \geq ||x'||_1 = ||x^U||_1 + \big(||x^W||_1 - (c_2^U - c'_2)\big) - (c'_1 + 2c'_2) = \mu^U(0) + c'_1 + c'_2 + ||x^W||_1 - (c_2^U + c'_1 + c'_2) = ||x^W||_1 + \mu^U(0) - c_2^U$. □

We finally prove (in a similar way as above) the main result in this paper.

Theorem 1. *For every pair $G = (V, E), b$ with $sw(G) \leq k$, we can solve* b-Matching *in $\mathcal{O}((k \log^2 k) \cdot (m + n) \cdot \log ||b||_1)$-time.*

Proof. Let $C_1, C_2, \ldots, C_s$, $s = sc(G)$ be the split components in any minimal split decomposition of G. Furthermore, let T be the corresponding split decomposition tree. It can be computed in linear-time [37]. We root T in C_1. For every $1 \leq i \leq s$, let T_i be the subtree of T that is rooted in C_i. If $i > 1$ then let $C_{p(i)}$ be its parent in T. By construction of T, the edge $\{C_{p(i)}, C_i\} \in E(T)$ corresponds to a split (U_i, W_i) of G, where $V(C_i) \cap V \subseteq U_i$. Let $G_{U_i} = (U_i \cup \{w_i\}, E_{U_i})$, $G_{W_i} = (W_i \cup \{u_i\}, E_{W_i})$ be the corresponding subgraphs of G. We can observe that T_i is a split decomposition tree of G_{U_i}, while $T \setminus T_i$ is a split decomposition tree of G_{W_i}.

Our algorithm proceeds in two main steps, with each step corresponding to a different traversal of the tree T.

Step 1: Computing the cardinality of the solution. Let $G_1 = G$ and let $G_i = G_{U_i}$ for every $i > 1$. Let $b^1 = b$ and, for every $i > 1$ let b^i be the restriction of b to U_i. We note that for any $i > 1$, b^i does not assign any capacity to the split marker vertex w_i. Up to adding a dummy isolated vertex w_1 to G_1, we assume that this above property holds for any i. Then, for any i, we compute the triple $(\mu^i(0), c_1^i, c_2^i)$ w.r.t. G_i, b^i and w_i (as defined in Proposition 1). In order to do so, we proceed by dynamic programming on the tree T, as follows. Let $C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ be the children of C_i in T. Every edge $\{C_i, C_{i_t}\} \in E(T)$, $1 \leq t \leq \ell$ also corresponds to a split (U_{i_t}, W_{i_t}) of G_i, where $V(C_{i_t}) \cap V(G_i) \subseteq U_{i_t}$. We name $w_{i_t} \in V(C_{i_t})$, $u_{i_t} \in V(C_i)$ the vertices added after the simple decomposition. Furthermore, let $(\mu^{i_t}(0), c_1^{i_t}, c_2^{i_t})$ be the triple corresponding to G_{i_t}, b^{i_t} and w_{i_t} (obtained by dynamic programming). We apply the reduction rule of Definition 2 — *i.e.*, we replace the split marker vertex u_{i_t} by the module $M_{i_t} = \{u_{i_t}^1, u_{i_t}^2, u_{i_t}^3\}$ where $u_{i_t}^1$ has capacity $c_1^{i_t}$ and the two adjacent vertices $u_{i_t}^2, u_{i_t}^3$ have equal capacity $c_2^{i_t}$. Doing so for every $1 \leq t \leq \ell$, we obtain

a pair H_i, b^{H_i} where H_i is a supergraph of C_i such that: $|V(H_i)| \leq 3|V(C_i)| \leq \max\{3k, 9\}$ and $|E(H_i)| \leq 9|E(C_i)| + |V(C_i)|$. Let us compute the triple $(\mu^{H_i}(0), c_1^i, c_2^i)$ corresponding to H_i, b^{H_i} and w_i. By Proposition 1 it can be done in time $\mathcal{O}(|V(H_i)||E(H_i)| \log^2 |V(H_i)| \log ||b||_1)$, that is in $\mathcal{O}((k \log^2 k) \cdot (|E(C_i)| + |V(C_i)|) \log ||b||_1)$.

Finally, by applying Proposition 2 for every split (U_{i_t}, W_{i_t}) we have:

$$\mu^i(0) = \mu^{H_i}(0) + \sum_{t=1}^{\ell} (\mu^{i_t}(0) - c_2^{i_t}).$$

Overall, this step takes total time $\mathcal{O}((k \log^2 k) \cdot \sum_i (|E(C_i)| + |V(C_i)|) \cdot \log ||b||_1) = \mathcal{O}((k \log^2 k) \cdot (n+m) \cdot \log ||b||_1)$. Furthermore, since $G_1 = G$, we have computed the maximum cardinality $\mu^1(0)$ of any b-matching of G.

Step 2: Computing an optimal solution. We compute a b-matching for the pair G, b that is of maximum cardinality $\mu^1(0)$, in two passes: the first one uses a top-to-bottom dynamic programming on T, and the second one uses a bottom-to-top dynamic programming on T.

The output of Step 2 is more general than what we did in Step 1, thus seemingly subsuming this previous step. However, Step 1 is needed in order to properly initialize the capacities of all the subgraphs considered during Step 2. Indeed, we consider the same pairs H_i, b^{H_i} as for Step 1. Each subgraph H_i contains some triples of vertices $u_{i_t}^1, u_{i_t}^2, u_{i_t}^3$, that are in one-to-one correspondence with the edges $\{C_i, C_{i_t}\}$ of the split decomposition tree. Their respective capacities were computed during Step 1 by bottom-to-top dynamic programming. Unlike for Step 1, the capacity of vertex w_i is also fixed (computed by top-to-bottom dynamic programming).

Specifically, for any i let $b_{w_i}^i$ be a fixed capacity for the split marker vertex w_i (if $i = 1$ then we set $b_{w_1}^1 = 0$; otherwise, for $i > 1$, $b_{w_i}^i$ is obtained by top-to-bottom dynamic programming). In what follows, we compute a maximum-cardinality b-matching for the pair G_i, b^i. For that we set $b_{w_i}^{H_i} = b_{w_i}^i$.

- We compute a maximum-cardinality b-matching x^i for the pair H_i, b^{H_i} such that both the saturation property and the symmetry property hold for every $1 \leq t \leq \ell$. By Lemma 8, it can be done in time $\mathcal{O}(|V(H_i)||E(H_i)| \log^2 |V(H_i)|)$, that is in $\mathcal{O}((k \log^2 k) \cdot (|E(C_i)| + |V(C_i)|))$.
- For every $1 \leq t \leq \ell$, let us define $b_{u_{i_t}}^i = deg_{x^i}(u_{i_t}^1) + 2 \cdot (deg_{x^i}(u_{i_t}^2) - x^i_{\{u_{i_t}^2, u_{i_t}^3\}})$. We merge M_{i_t} into the original split marker vertex u_{i_t}. Furthermore, we assign the capacity $b_{u_{i_t}}^i$ to u_{i_t}, and we update the b-matching x^i such that, for every $v \in N_{H_i}(M_{i_t})$, we have $x^i_{\{v,u_{i_t}\}} = x^i_{\{v,u_{i_t}^1\}} + x^i_{\{v,u_{i_t}^2\}} + x^i_{\{v,u_{i_t}^3\}}$. Doing so, we transform x^i into a b-matching for C_i, b^i. Recall that since the symmetry property holds, we have $deg_{x^i}(u_{i_t}^2) = deg_{x^i}(u_{i_t}^3)$. In particular, we have after the transformation that:

$$\begin{aligned}\sum_{v \in N_{H_i}(M_{i_t})} x^i_{\{v,u_{i_t}\}} &= deg_{x^i}(u_{i_t}^1) + (deg_{x^i}(u_{i_t}^2) - x^i_{\{u_{i_t}^2, u_{i_t}^3\}}) + (deg_{x^i}(u_{i_t}^3) - x^i_{\{u_{i_t}^2, u_{i_t}^3\}}) \\ &= deg_{x^i}(u_{i_t}^1) + 2 \cdot (deg_{x^i}(u_{i_t}^2) - x^i_{\{u_{i_t}^2, u_{i_t}^3\}}) \\ &= b_{u_{i_t}}^i.\end{aligned}$$

  As a result, in x^i, all vertices $u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}$ are saturated.

- For every $1 \leq t \leq \ell$ we set $b^{i_t}_{w_{i_t}} = b^i_{u_{i_t}}$ and then we compute a maximum-cardinality b-matching x^{i_t} for the pair G_{i_t}, b^{i_t}. Note that since the saturation property holds, we have either $b^i_{u_{i_t}} \leq c^{i_t}_1$ or $b^i_{u_{i_t}} = c^{i_t}_1 + 2 \cdot d_{i_t}$ where $d_{i_t} = deg_{x^i}(u^2_{i_t}) - x^i_{\{u^2_{i_t}, u^3_{i_t}\}} \leq c^{i_t}_2$. Then, we can deduce from Proposition 1 that any decrease of $b^{i_t}_{w_{i_t}}$ $(= b^i_{u_{i_t}})$ would result in a decrease of the maximum-cardinality of a b-matching for the pair G_{i_t}, b^{i_t}. As a result, in x^{i_t}, the split marker vertex w_{i_t} is saturated.
- By applying the routine of Lemma 10 for every incident split (U_{i_t}, W_{i_t}) we merge x^i with all the x^{i_t}'s until we obtain a b-matching for G_i, b^i (bottom-to-top dynamic programming). It takes time at most $K \cdot (\varphi(G_i) - \sum_{t=1}^{l} \varphi(G_{i_t}))$, for some universal constant K.

Finally, the above b-matching is of maximum cardinality, that follows from Proposition 2 (applied for every incident split). Overall, this second step of the algorithm takes total time $\mathcal{O}(\varphi(G)) + \mathcal{O}((k \log^2 k) \cdot \sum_i (|E(C_i)| + |V(C_i)|)) = \mathcal{O}((k \log^2 k) \cdot (n + m))$. □

Setting $b_v = 1$ for every $v \in V$, we obtain the following implication of Theorem 1:

Corollary 1. *For every graph $G = (V, E)$ with $sw(G) \leq k$, we can solve* Maximum-Cardinality Matching *in $\mathcal{O}((k \log^2 k) \cdot (m + n) \cdot \log n)$-time.*

5. Open questions

We presented an algorithm for solving b-Matching on distance-hereditary graphs, and more generally on any graph with bounded split-width. In contrast to our result, we stress that as already noticed in [29], Maximum-Weight Matching cannot be solved faster on complete graphs, and so, on distance-hereditary graphs, than on general graphs. An interesting open question would be to know whether b-Matching can be solved in *linear* time on bounded split-width graphs. In a companion paper [15], we prove with a completely different approach that Maximum-Cardinality Matching can be solved in $\mathcal{O}(n+m)$-time on distance-hereditary graphs. However, it is not clear to us whether similar techniques can be used for bounded split-width graphs in general.

References

[1] H.-J. Bandelt and H. Mulder. Distance-hereditary graphs. *J. of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[2] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957.

[3] J. A. Bondy and U. S. R. Murty. *Graph theory*. Grad. Texts in Math., 2008.

[4] H. Bunke. Graph matching: Theoretical foundations, algorithms, and applications. In *Proc. Vision Interface*, volume 2000, pages 82–88, 2000.

[5] M. Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In *International Symposium on Algorithms and Computation*, pages 146–155. Springer, 1996.

[6] P. Charbit, F. De Montgolfier, and M. Raffinot. Linear time split decomposition revisited. 26(2):499–514, 2012.

[7] D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Transactions on Algorithms (TALG)*, 15(3):1–57, 2019.

[8] W. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, 1982.

[9] E. Dahlhaus and M. Karpinski. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics*, 84(1-3):79–91, 1998.

[10] E. Dekel and S. Sahni. A parallel matching algorithm for convex bipartite graphs and applications to scheduling. *Journal of Parallel and Distributed Computing*, 1(2):185–205, 1984.

[11] S. Dong, Y. Lee, and G. Ye. A Nearly-Linear Time Algorithm for Linear Programs with Small Treewidth: A Multiscale Representation of Robust Central Path. 2021. To appear in STOC'21. Available online at `https://arxiv.org/abs/2011.05365`.

[12] F. Dragan. On greedy matching ordering and greedy matchable graphs. In *WG'97*, volume 1335 of *LNCS*, pages 184–198. Springer, 1997.

[13] F. Dragan and F. Nicolai. LexBFS-orderings of distance-hereditary graphs with application to the diametral pair problem. *Discrete Applied Mathematics*, 98(3):191–207, 2000.

[14] G. Ducoffe and A. Popa. The b-Matching Problem in Distance-Hereditary Graphs and Beyond. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[15] G. Ducoffe and A. Popa. The Use of a Pruned Modular Decomposition for Maximum Matching Algorithms on Some Graph Classes. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[16] J. Edmonds. Paths, trees, and flowers. *Canadian J. of mathematics*, 17(3):449–467, 1965.

[17] J. Edmonds and E. Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial structures and their applications*. Citeseer, 1970.

[18] F. Fomin, D. Lokshtanov, S. Saurabh, M. Pilipczuk, and M. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms (TALG)*, 14(3):1–45, 2018.

[19] J.-L. Fouquet, V. Giakoumakis, and J.-M. Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *International J. of Foundations of Computer Science*, 10(04):513–533, 1999.

[20] J.-L. Fouquet, I. Parfenoff, and H. Thuillier. An $O(n)$-time algorithm for maximum matching in P_4-tidy graphs. *Information processing letters*, 62(6):281–287, 1997.

[21] H. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *STOC'83*, pages 448–456. ACM, 1983.

[22] H. Gabow. Data Structures for Weighted Matching and Extensions to b-matching and f-factors. *ACM Transactions on Algorithms (TALG)*, 14(3):1–80, 2018.

[23] A. C. Giannopoulou, G. B. Mertzios, and R. Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science*, 689:67–95, 2017.

[24] F. Glover. Maximum matching in a convex bipartite graph. *Naval Research Logistics (NRL)*, 14(3):313–316, 1967.

[25] M. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *International J. of Foundations of Computer Science*, 11(03):423–443, 2000.

[26] J. Hopcroft and R. Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[27] Y. Iwata, T. Ogasawara, and N. Ohsaka. On the power of tree-depth for fully polynomial FPT algorithms. In *STACS'18*, 2018.

[28] R. Karp and M. Sipser. Maximum matching in sparse random graphs. In *FOCS'81*, pages 364–375. IEEE, 1981.

[29] S. Kratsch and F. Nelles. Efficient and adaptive parameterized algorithms on modular decompositions. In *ESA'18*. LIPIcs, 2018. To appear.

[30] Y. Liang and C. Rhee. Finding a maximum matching in a circular-arc graph. *Information processing letters*, 45(4):185–190, 1993.

[31] L. Lovász and M. Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.

[32] G. Mertzios, A. Nichterlein, and R. Niedermeier. The power of linear-time data reduction for maximum matching. In *MFCS'17*, pages 46:1–46:14, 2017.

[33] G. Mertzios, A. Nichterlein, and R. Niedermeier. A Linear-Time Algorithm for Maximum-Cardinality Matching on Cocomparability Graphs. *SIAM Journal on Discrete Mathematics*, 32(4):2820–2835, 2018.

[34] S. Micali and V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *FOCS'80*, pages 17–27. IEEE, 1980.

[35] M. Penn and M. Tennenholtz. Constrained multi-object auctions and b-matching. *Information Processing Letters*, 75(1-2):29–34, 2000.

[36] W. Pulleyblank. Matchings and extensions. *Handbook of combinatorics*, 1:179–232, 1995.

[37] M. Rao. Solving some NP-complete problems using split decomposition. *Discrete Applied Mathematics*, 156(14):2768–2780, 2008.

[38] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC'13*, pages 515–524. ACM, 2013.

[39] M. Tennenholtz. Tractable combinatorial auctions and b-matching. *Artificial Intelligence*, 140(1-2):231–243, 2002.

[40] P. Tseng and Z.-Q. Luo. On computing the nested sums and infimal convolutions of convex piecewise-linear functions. *Journal of Algorithms*, 21(2):240–266, 1996.

[41] W. Tutte. A short proof of the factor theorem for finite graphs. *Canad. J. Math*, 6(1954):347–352, 1954.

[42] M.-S. Yu and C.-H. Yang. An $O(n)$-time algorithm for maximum matching on cographs. *Information processing letters*, 47(2):89–93, 1993.

[43] R. Yuster. Maximum matching in regular and almost regular graphs. *Algorithmica*, 66(1):87–92, 2013.

[44] R. Yuster and U. Zwick. Maximum matching in graphs with an excluded minor. In *Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 108–117. Society for Industrial and Applied Mathematics, 2007.