The b-Matching Problem in Distance-Hereditary Graphs and Beyond

Guillaume Ducoffe
ICI – National Institute for Research and Development in Informatics, Bucharest, Romania
The Research Institute of the University of Bucharest ICUB, Bucharest, Romania
guillaume.ducoffe@ici.ro

Alexandru Popa
University of Bucharest, Bucharest, Romania
ICI – National Institute for Research and Development in Informatics, Bucharest, Romania
alexandru.popa@fmi.unibuc.ro

Abstract

We make progress on the fine-grained complexity of Maximum-Cardinality Matching on graphs of bounded *clique-width*. Quasi linear-time algorithms for this problem have been recently 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 improve the state of the art (Dragan, WG'97) and 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.

2012 ACM Subject Classification Mathematics of computing → Graph theory, Theory of computation → Design and analysis of algorithms

Keywords and phrases maximum-cardinality matching, b-matching, FPT in P, split decomposition, distance-hereditary graphs

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.30

Related Version Full proofs can be found in our technical report [9], `https://arxiv.org/abs/1804.09393`.

Funding This work was supported by the Institutional research programme PN 1819 "Advanced IT resources to support digital transformation processes in the economy and society - RESINFO-TD" (2018), project PN 1819-01-01 "Modeling, simulation, optimization of complex systems and decision support in new areas of IT&C research", funded by the Ministry of Research and Innovation, Romania.

1 Introduction

The Maximum-Cardinality Matching problem takes as input a graph $G = (V, E)$ and it asks for a subset F of pairwise disjoint edges of maximum cardinality. This is a fundamental problem with a wide variety of applications. Hence, the computational complexity of Maximum-Cardinality Matching has been extensively studied in the literature. For instance, this was the first problem shown to be solvable in polynomial-time [11]. Currently, the best-known algorithms for this problem run in $\mathcal{O}(m\sqrt{n})$-time on n-vertex m-edge graphs [22]. 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 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, 8, 17]. In particular, we can solve Diameter in linear-time on this class of graphs [8]. For the latter problem on general graphs, a conditional *quadratic* lower-bound has been proved in [24]. 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 [26] and (tent,hexahedron)-free distance-hereditary graphs [7]. 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.

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 [6], 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 [23].

Many NP-hard problems can be solved in polynomial time on bounded split-width graphs (*e.g.*, Graph Coloring, see [23]). Recently, with Coudert, we designed FPT algorithms for polynomial problems when parameterized by split-width [5]. 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 [5] 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 [5] 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 $\pi(G) \leq k$. 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 [5, 16, 19, 20, 21]. We stress that Maximum-Cardinality Matching has been proposed in [21] 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* [17]. In [5], 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 [13, 19]. Turning our attention on denser graph classes of bounded clique-width, we proved in [5] 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 [23], whereas split-width is upper-bounded by modular-width [5]. 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 [5], 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 [12]. Roughly, every vertex v is assigned some input capacity b_v, and the goal is to compute 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 a piecewise linear function 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 [5]. 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 17). 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 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.

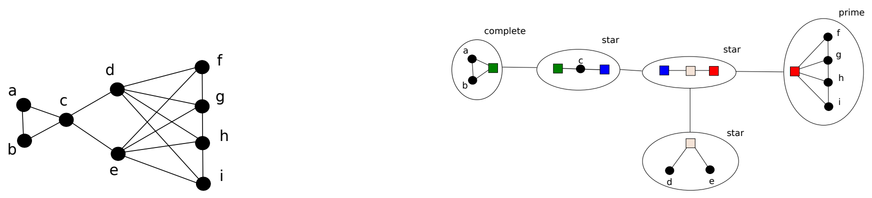

Figure 1 A graph and its split decomposition. Split marker vertices that correspond to a same simple decomposition are identified by two rectangles with the same color.

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. Due to space restrictions, some of the proofs are omitted. Full proofs can be found in our technical report [9].

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 neighbourhood 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 neighbourhood 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

Let a *split* in a graph $G = (V, E)$ be a partition $V = U \cup W$ such that: $\min\{|U|, |W|\} \geq 2$; and there is a complete join between the vertices of $N_G(U)$ and $N_G(W)$. A *simple decomposition* of G takes as input a 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 $N_{G_U}(w) = U$ and $N_{G_W}(u) = W$. 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.

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. Degenerate graphs are such that every bipartition of their vertex-set is a split. They are exactly the complete graphs and the stars [6]. A graph is prime for split decomposition if it has no split. We can define the following two types 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 [4].
- **Minimal split decomposition.** A split-decomposition is *minimal* if all the subgraphs obtained are prime. A minimal split-decomposition can be computed from the canonical split-decomposition in linear-time [6]. Doing so, we avoid handling with the particular cases of stars and complete graphs in our algorithms. The set of prime graphs in any minimal split decomposition is unique up to isomorphism [6].

For instance, the split decomposition of Fig. 1 is both minimal and canonical.

▶ **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 [23] 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 [23].

2.2 Matching problems

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

▶ **Problem 2** (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 [22]. 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 3** (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 4** (Hopcroft-Karp, [18]). *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 5** (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 6** (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 7** ([14, 15]). *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 [25]). 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 [12]. Independently from this work[1], the authors in [20] proposed a new Maximum-Cardinality Matching algorithm

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

on graphs of bounded modular-width that is also based on a reduction to b-MATCHING. Unlike this work, the algorithm of [20] 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 [20] – 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 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 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.

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 (Proposition 11). We start by introducing some useful lemmata.

▶ **Lemma 8.** $\mu(t+1) - \mu(t) \leq 1$.

▶ **Lemma 9.** *If $\mu(t+2) = \mu(t)$ then we have $\mu(t+i) = \mu(t)$ for every $i \geq 0$.*

▶ **Lemma 10.** *If $\mu(t+1) = \mu(t)$ then we have $\mu(t+3) = \mu(t+2)$.*

These above results are obtained by studying vertex-disjoint augmenting paths in some "expanded graphs" $G_{b,t}$ (cf. Lemmata 3 and 4).

▶ **Proposition 11.** *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.

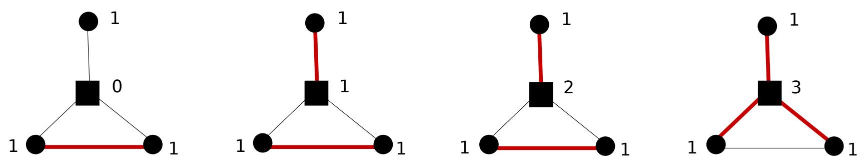

Figure 2 An example with $(\mu(0), c_1, c_2) = (1, 1, 1)$. Vertices are labeled with their capacity. Thin and bold edges have respective weights 0 and 1.

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 neighbours). Furthermore, by Lemma 8 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 10, 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 9 $\mu(c_1 + 2i) < \mu(c_1 + 2(i + 1))$ (since otherwise $t_{\max} \leq c_1 + 2i$). By Lemma 10 we have $\mu(c_1 + 2i) = \mu(c_1 + 2i + 1)$. Finally, by Lemma 8 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 7, 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 17). 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 12. Finally, starting from the root component C_1, we compute a maximum-cardinality b-matching of G, b by reverse 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.

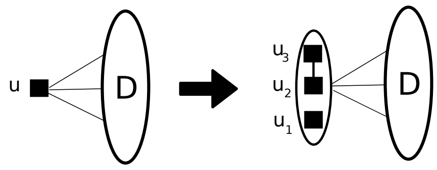

Figure 3 The reduction of Definition 12.

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* [2], as follows:

▶ **Definition 12.** 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 11 w.r.t. G_U, b^U and w.
- 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 13.** *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 12. 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 13 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 17.

▶ **Lemma 14.** *Let x be a b-matching for G, b. There exists a b-matching x^W for H_W, b^W such that $||x^W||_1 \geq ||x||_1 + c^U_2 - \mu^U(0)$.*

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 13 (as well as Theorem 17).

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. Let C_i be a split component in a given split decomposition of G. The subgraph C_i is obtained from a sequence of simple decompositions. For a given subsequence of the above simple decompositions (corresponding to the edges between C_i and

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

its children in T) we apply the reduction rule of Definition 12. Doing so, we obtain a pair H_i, b^i with H_i being a supergraph of C_i obtained by replacing some 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 $u^2_{i_t}, u^3_{i_t}$ are adjacent and they have the same capacity.

We seek for a maximum-cardinality b-matching x^i for the pair H_i, b^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 $deg_{x^i}(u^1_{i_t}) < c^1_{i_t}$ 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 15.** *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^i such that both the saturation property and the symmetry property hold for every M_{i_t}, $1 \leq t \leq \ell$.*

4.3 Merging the partial solutions together

Finally, before we can describe our main algorithm (Theorem 17) 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, this merging stage can be solved with a greedy algorithm.

▶ **Lemma 16.** *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 $deg_{x^U}(w) = deg_{x^W}(u) = d$.*

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 [3].

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

Overall, since there are at most $n - 2$ components in any minimal split decomposition of G [23], the merging stages take total time $\mathcal{O}(\varphi(G)) = \mathcal{O}(n + m)$.

4.4 Main result

We are now ready to prove Proposition 13. This algorithmic proof is the cornerstone of our main result.

Proof of Proposition 13. We have $||x^W||_1 \geq ||x||_1 - \mu^U(0) + c^U_2$ by Lemma 14. 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 15, 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

[3] We recall that the set of prime graphs in any minimal split decomposition is unique up to isomorphism [23].

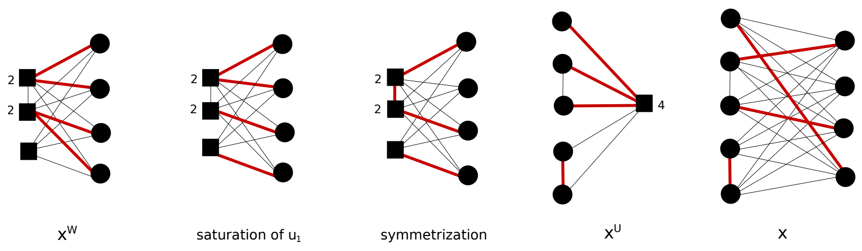

Figure 4 The construction of x'. Vertices with capacity greater than 1 are labeled with their capacity. Thin and bold edges have respective weights 0 and 1.

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 $c_1' = c_1^U$. Furthermore, we observe that u_2 and u_3 must be saturated (otherwise, we could increase the cardinality of the b-matching by setting $x^W_{\{u_2,u_3\}} = c_2^U - c_2'$). Therefore, we get:

$$||x^W||_1 = \mu^W + c_1' + 2c_2' + (c_2^U - c_2') = \mu^W + c_1' + c_2' + c_2^U.$$

We define $b_u^W = b_w^U = c_1' + 2c_2'$. Then, we proceed as follows (see Fig. 4 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 $deg_{x^W}(u) = b_u^W = c_1' + 2c_2'$. Furthermore, the cardinality of the b-matching has decreased by $x^W_{\{u_2,u_3\}} = c_2^U - c_2'$.
- 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 11: $||x^U||_1 = \mu^U(c_1' + 2c_2') = \mu^U(0) + c_1' + c_2'$; 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 16. Doing so, we get $||x||_1 \geq ||x'||_1 = ||x^U||_1 + \left(||x^W||_1 - (c_2^U - c_2')\right) - (c_1' + 2c_2') = \mu^U(0) + c_1' + c_2' + ||x^W||_1 - (c_2^U + c_1' + c_2') = ||x^W||_1 + \mu^U(0) - c_2^U$. ◀

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

▶ **Theorem 17.** *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.*

Setting $b_v = 1$ for every $v \in V$, we obtain the following implication of Theorem 17:

▶ **Corollary 18.** *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 [20], 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 [10], 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 P. Charbit, F. De Montgolfier, and M. Raffinot. Linear time split decomposition revisited. *SIAM J. on Discrete Mathematics*, 26(2):499–514, 2012.

5 D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *SODA'18*, pages 2765–2784. SIAM, 2018.

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

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

8 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.

9 G. Ducoffe and A. Popa. A quasi linear-time b-Matching algorithm on distance-hereditary graphs and bounded split-width graphs. Technical Report arXiv:1804.09393, arXiv, 2018.

10 G. Ducoffe and A. Popa. The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes. In *ISAAC*, 2018. To appear.

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

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

13 F. Fomin, D. Lokshtanov, M. Pilipczuk, S. Saurabh, and M. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. In *SODA'17*, pages 1419–1432. SIAM, 2017.

14 H. Gabow. An Efficient Reduction Technique for Degree-Constrained Subgraph and Bidirected Network Flow Problems. In *STOC'83*, pages 448–456. ACM, 1983.

15 H. Gabow. Data Structures for Weighted Matching and Extensions to *b*-matching and *f*-factors. *arXiv*, 2016. `arXiv:1611.07541`.

16 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.

17 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.

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

19 Y. Iwata, T. Ogasawara, and N. Ohsaka. On the Power of Tree-Depth for Fully Polynomial FPT Algorithms. In *STACS'18*, 2018.

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

21 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.

22 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.

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

24 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.

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

26 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.