Maximum Matching in almost linear time on graphs of bounded clique-width

Guillaume Ducoffe

Maximum Matching in almost linear time on graphs of bounded clique-width

Guillaume Ducoffe
National Institute for Research and Development in Informatics, Romania
University of Bucharest, Romania

Abstract

Recently, independent groups of researchers have presented algorithms to compute a maximum matching in $\tilde{\mathcal{O}}(f(k) \cdot (n+m))$ time, for some computable function f, within the graphs where some clique-width upper bound is at most k (*e.g.*, tree-width, modular-width and P_4-sparseness). However, to the best of our knowledge, the existence of such algorithm within the graphs of bounded clique-width has remained open until this paper. Indeed, we cannot even apply Courcelle's theorem to this problem directly, because a matching cannot be expressed in MSO_1 logic.

Our first contribution is an almost linear-time algorithm to compute a maximum matching in any bounded clique-width graph, being given a corresponding clique-width expression. It can be used to also compute the Edmonds-Gallai decomposition. For that, we do apply Courcelle's theorem, but in order to compute the cardinality of a maximum matching rather than the matching itself, via the classic Tutte-Berge formula. To obtain with this approach a maximum matching, we need to combine it with a recursive dissection scheme for bounded clique-width graphs based on the existence of balanced edge-cuts with bounded neighbourhood diversity, and with a distributed version of Courcelle's theorem (Courcelle and Vanicat, *DAM 2016*) – of which we present here a slightly stronger version than the standard one in the literature – in order to evaluate the Tutte-Berge formula on various subgraphs of the input.

Finally, for the bipartite graphs of clique-width at most k, we present an alternative $\tilde{\mathcal{O}}(k^2 \cdot (n+m))$-time algorithm for the problem. The algorithm is randomized and it is based on a completely different approach than above: combining various reductions to matching and flow problems on bounded tree-width graphs with a very recent result on the parameterized complexity of linear programming (Dong et. al., *STOC'21*). Our results for bounded clique-width graphs extend many prior works on the complexity of Maximum Matching within cographs, distance-hereditary graphs, series-parallel graphs and other subclasses.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Maximum Matching; Maximum b-matching; Clique-width; Tree-width; Courcelle's theorem; FPT in P.

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.15

1 Introduction

For any undefined graph terminology, see [7, 23]. Throughout the paper, for any graph $G = (V, E)$, let $n := |V|$ be its *order* (number of vertices) and $m := |E|$ be its *size* (number of edges). Recall that a *matching* in a graph is a set of pairwise end-disjoint edges. A *maximum matching* is one of maximum cardinality. The *matching number* of G, denoted by $\nu(G)$, is the cardinality of a maximum matching of G. Matchings (possibly, with additional constraints) are ubiquitous in scheduling, markets, resource allocation schemes and even chemistry [68]. We refer to [44, 55] for a compendium of matching problems and their applications.

This paper is about the (parameterized) complexity of Maximum Matching in graphs. Unsurprisingly, a lot of research in Computer Science has been devoted to this question. The first polynomial-time algorithm for Maximum Matching was proposed by Edmonds [30]. Later, Micali and Vazirani presented the state-of-the-art $\mathcal{O}(m\sqrt{n})$-time algorithm for this problem [60], that has remained unchallenged since forty years. We here study Maximum Matching in the context of "Fine-Grained Complexity in P" (see [72] for a survey of this blossoming field). Specifically, *can a maximum matching be computed in (almost) linear time?* There are a few reasons to believe that it is indeed the case. For instance, unlike for the diameter problem and other fundamental graph problems, for which over the last decades, conditional superlinear lower bounds were obtained, it is known [8] that proving such lower bound for Maximum Matching would falsify the so-called Nondeterministic Strong Exponential Time Hypothesis (NSETH). Furthermore, computing a maximum matching is related to Maximum Flow [71], that is sometimes conjectured to be solvable in linear time.

The idea of using tools and concepts from parameterized complexity in the context of polynomial-time solvable problems has been scarce [48]. In part motivated by the recent "SETH-hardness" results, and other conditional lower bounds for such problems [73], a richer theory of "FPT in P" has started to emerge recently [1, 5, 32, 36, 45]. In its simplest form, the former is about the existence of $\mathcal{O}(f(k) \cdot (n+m)^{1+o(1)})$-time algorithms for various graph problems when some fixed parameter is at most k [1]. As far as we are concerned here, such running times were obtained in [13, 29, 28, 36, 45, 49, 50, 51, 59] for Maximum Matching, for different parameterizations. For instance, for the graphs of *tree-width* at most k, Fomin et. al [36] presented a randomized $\tilde{\mathcal{O}}(k^4 \cdot n)$-time algorithm for computing a maximum matching [2]. This was later improved by Iwata et. al. [50], who designed a deterministic $\tilde{\mathcal{O}}(k^2 \cdot n)$-time algorithm for that problem. – We recall the definition of tree-width in Sec. 2.2. – Remarkably, the parameterized study of Maximum Matching has led to the development of many nice techniques in this area, which brought Mertzios et. al. [59] to nickname the problem the "drosophilia" of the study of the FPT algorithms in P.

Clique-width is one of the most studied graph parameters. It is a rough estimate of the closeness of a graph to a cograph (*a.k.a.*, P_4-free graph). We refer to Sec. 2.1 for a formal definition. Note that unlike for the tree-width, there exist dense graphs of bounded clique-width (*e.g.*, the complete graphs and the complete bipartite graphs). The applications of clique-width to NP-hard problems, including Courcelle's theorem [15] and some general algorithmic frameworks [31], are now rather well understood [33, 34, 35]. However, the study of its applications to polynomial-time solvable problems is comparatively much more

[1] More generally, the goal is, for some problem solvable in $\mathcal{O}(m^{q+o(1)})$ time on arbitrary m-edge graphs, to design an $\mathcal{O}(f(k)m^{p+o(1)})$-time algorithm, for some $p < q$, within the class of graphs where some fixed parameter is at most k.

[2] The $\tilde{\mathcal{O}}()$ notation suppresses poly-logarithmic factors.

recent and, so far, limited to cycle problems [5, 13] and distance problems [13, 16, 27, 52]. Parameterized almost linear-time algorithms for Maximum Matching are known for the important subclasses of bounded tree-width graphs [36, 50], graphs of bounded modular-width [13, 51], and some others [13, 28, 29]. However, as far as we know, the complexity of this problem on bounded clique-width graphs has been open until this article. Indeed we stress that, even allowing a super-polynomial dependency on the clique-width in the running time, the existence of an almost linear-time (parameterized) algorithm for Maximum Matching does not follow from Courcelle's theorem, because a matching cannot be expressed in MSO_1 logic. This is in sharp contrast with bounded tree-width graphs, for which we can apply Courcelle's theorem for the stronger MSO_2 logic (allowing quantification over subsets of edges), and so, in particular in order to express a matching [18]. – We refer to Sec. 3 for a reminder about MSO logic. – Furthermore if we consider the related problem Maximum-Weight Matching, then it has been observed [51] that it is as hard on bounded clique-width weighted graphs as on general weighted graphs under $\mathcal{O}(n^2)$-time reductions. Again, this differs from the case of bounded tree-width graphs, for which an $\tilde{\mathcal{O}}(k^2n)$-time algorithm also exists for this problem [50].

Beyond the study of the FPT algorithms in P, it also makes sense to study Maximum Matching on restricted graph classes, both as a way to better understand the hard instances for this problem, and to better model some of its real-life applications (see [46] for an example of the latter). In this respect, a considerable amount of positive results have been proved [9, 22, 26, 38, 37, 46, 54, 58, 61, 74, 76, 75]. Many such classes, starting from the cographs [20], are known to have bounded clique-width. Therefore, having at hands an almost linear-time algorithm for Maximum Matching on bounded clique-width graphs, one can unify and generalize many prior works in this area.

1.1 Our results

Recall that a graph has clique-width at most k if and only if it admits a *k-expression* [20]. Such a k-expression can be computed in linear time on many interesting subclasses of bounded clique-width graphs [3]: ranging from cographs [20], switched cographs [11], distance-hereditary graphs [47], $(q, q-3)$-graphs [57], and graphs of either bounded tree-width [12], modular-width [20] or split-width [65].

Hereafter, we use the notation $\tilde{\mathcal{O}}_{x_1,x_2,\ldots,x_t}(n+m)$ for a running time in $\tilde{\mathcal{O}}(f(x_1, x_2, \ldots, x_t)\cdot(n+m))$, for some computable function f. The following is our first main result in the paper:

▶ **Theorem 1.** *Given a graph G and a corresponding k-expression, one can compute a maximum matching for G in deterministic $\tilde{\mathcal{O}}_k(n+m)$ time.*

To the best of our knowledge, this is the first almost linear-time algorithm for Maximum Matching on bounded clique-width graphs. The $\tilde{\mathcal{O}}_k()$ notation hides huge constants in k due to our use of Courcelle's theorem. Indeed, while we cannot express a matching in MSO_1 logic, we can write a *Counting* MSO_1 formula in order to evaluate the matching number (Theorem 6). For that, we use the well-known *Tutte-Berge* formula [6]. This alone does not lead to an efficient computation of a maximum matching, but only of its size. However, by carefully evaluating our formula for the matching number on various subgraphs, obtained by removing specific vertex- and edge-subsets, one can compute a maximum matching incrementally. A

[3] So far, the best-known approximation algorithms for clique-width run in $\mathcal{O}(n^3)$-time, that is slower than the state-of-the-art algorithm for Maximum Matching [62]

similar approach also works for computing the *Edmonds-Gallai decomposition* [30, 42, 43], which somehow encodes the structure of all the maximum matchings in a graph (Theorem 8). The main difficulty here is that the number of subgraphs on which we need to evaluate our formula is not constant. Thus, applying Courcelle's theorem to each subgraph separately would result in a super-linear running time. We overcome this issue by using a distributed version of this theorem [17]. In doing so, after we computed the matching number of a bounded clique-width graph G in almost linear time, it becomes possible to evaluate our formula on any subgraph H in time roughly proportional to the number of basic operations needed to obtain H from G.

It seems that improving the dependency on k in the running time will require new techniques. Our second main result is that it can be done for *bipartite* graphs of bounded clique-width:

▶ **Theorem 2.** *Given a bipartite graph G and a corresponding k-expression, one can compute a maximum matching for G in randomized $\tilde{\mathcal{O}}(k^2 \cdot (n+m))$ time.*

Let us sketch below the main lines of our approach toward proving Theorem 2. We first reduce Maximum Matching on bounded clique-width graphs to a related problem on the graphs of bounded *tree-width*. The reduction preserves the property for a graph to be bipartite. Its intuition goes as follows. Roughly, graphs of bounded tree-width can be recursively disconnected by some small balanced vertex-separators. By comparison, graphs of bounded clique-width can be recursively disconnected by some balanced edge-cuts of small "neighbourhood diversity" (partitionable in a small number of complete joins) [19]. To reduce to the bounded tree-width case, we propose a transformation of edge-cuts of small neighbourhood diversity into small vertex-separators (Sec. 5.1). The transformation forces us to deal with a more general problem than Maximum Matching, sometimes called Maximum b-Matching and well-studied on its own [4, 40, 41, 56, 63, 64]. We thus exchange Maximum Matching for a more complex problem, but on a structurally simpler graph class. Furthermore, because we restrict ourselves to bipartite graphs, we can solve Maximum b-Matching as a linear program. To the matrix representation of any such linear program, one can associate various graphs. Then, it becomes possible to define the tree-width of a linear program. In [36], Fomin et. al. asked whether all linear programs of bounded tree-width could be solved in almost linear time. Very recently, Dong et. al. gave a positive answer [25]. – This is where we need randomization. – We apply this nice result to the problem Maximum b-Matching within bipartite graphs. Here, some final technicalities arise due to the algorithm of Dong et. al. only outputting an approximate fractional b-matching whereas we aim at computing an exact integral solution. This can be overcome by using the close connection between Maximum Flow and Maximum b-Matching on bipartite graphs, along with a nice result from Madry to apply rounding to a fractional flow [56].

1.2 Related work

The are several meta-theorems deduced from Courcelle's theorem in the literature. Indeed, Courcelle's approach not only applies to decision problems, but also to counting [14] and optimization problems [15]. We actually use in our proof the optimization version of his theorem. Applications to the design of distance-labelling schemes were proposed in [17], and later refined in [16, 27] using alternative techniques. However, insofar most applications of Courcelle's theorem are about NP-hard problems. Indeed, Abboud et. al. [1] observed that its use leads to huge dependencies on the parameter involved, that can often be sharpened by preferring other techniques (their observation, on the other hand, also remains valid for

NP-hard problems). What we find intriguing in our case is, first, the nontrivial use we need to make of Courcelle's theorem for a polynomial-time solvable problem, and second, that we currently do not see any other way to obtain a quasi linear-time algorithm for Maximum Matching on the bounded clique-width graphs. This is evidence, we believe, that Courcelle's theorem could help in expanding the nascent field of "FPT in P".

The proof of our Theorem 1 also has several aspects that, we think, are equally intriguing. For one, we avoid computing augmenting paths, and we do not need the Tutte matrix [70] either. Both concepts are the cornerstone of almost all maximum matching algorithms in the literature. To the best of our knowledge, our result is one of the very first algorithmic applications of the Tutte-Berge formula. The latter also got used in [8], but the algorithm in this related work was non-deterministic. Our repeated use of this formula in order to compute a maximum matching is not unlike the celebrated result of Anari and Vazirani that reduces the efficient parallel computation of such matching to the design of an oracle for a decision version of the problem [3]. Nevertheless, both results have fairly different proofs.

About Theorem 2, we note that different reductions from Maximum Matching to Maximum b-Matching have already been considered for graphs of bounded modular-width [51] or bounded split-width [28], that are subclasses of bounded clique-width graphs. However, from the algorithmic point of view, the instances of Maximum b-Matching outputted by these former reductions are of bounded size, a much more restricted case than bounded tree-width. To our best knowledge, the Maximum b-Matching problem has only been solved in almost linear time on subclasses of graphs of tree-width at most two [29]. We left open the parameterized complexity of Maximum b-Matching within the graphs of bounded tree-width. For general graphs, the so-called "Russian method" starts from the linear relaxation of this problem (written as an integer program) and it repeatedly adds "blossom constraints" that are violated by the current solution until it finds an optimal integral outcome [63]. These blossom constraints are deduced from the good characterization of the b-matching polytope by Edmonds and Pulleyblank [64]. It seems, however, that a super-linear (but polynomial) number of linear programs needs to be solved on general graphs. See also Anstee [4] and Gabow [40] for alternative algorithms.

1.3 Organization of the paper

In Sec. 2, we introduce some basic notations and terminology, as well as the two graph parameters considered in this article. Sec. 3 is devoted to Courcelle's theorem for bounded clique-width graphs. We need a distributed version of this theorem, for optimization functions, of which a proof by Courcelle and Vanicat can be found in [17] but, unfortunately, for a more restricted setting than what we need. While it is not excessively difficult to check that Courcelle and Vanicat's proof indeed works in the broader setting that is here needed, the proof is fairly long and it has several intermediate steps, which is why we found it better to write it down almost completely. Then, in Sec. 4, we apply this result in order to compute the matching number, a corresponding maximum matching, and the Edmonds-Gallai decomposition, within bounded clique-width graphs. Sec. 5 is devoted to the proof of Theorem 2. We then conclude in Sec. 6.

2 Preliminaries

First we complete the basic graph terminology given in Sec. 1. By a graph, we mean a finite, simple, unweighted undirected graph. Let $G = (V, E)$ be such a graph. The (open) *neighbourhood* of a vertex v is defined as $N_G(v) := \{u \in V \mid uv \in E\}$. Its *closed neighbourhood*

is defined as $N_G[v] := N_G(v) \cup \{v\}$. Let $d_G(v) := |N_G(v)|$ be the *degree* of vertex v. Similarly, for a vertex-subset S, let $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ denote its (open) neighbourhood and let $N_G[S] := S \cup N_G(S)$ denote its closed neighbourhood. We sometimes omit the subscript if the graph G is clear from the context.

2.1 Clique-width

A *k-labeled graph* is a triple $G = (V, E, \ell)$ where $\ell : V \to \{1, 2, \ldots, k\}$ is called a labeling function. A k-expression is an algebraic expression where the four allowed operations are:

- $i(v)$: we add a new isolated vertex with label $\ell(v) = i$;
- $G_1 \oplus G_2$: we make the disjoint union of two k-labeled graphs;
- $\eta(i, j)$: we add a join (complete bipartite subgraph) between all vertices with label i and all vertices with label j;
- $\rho(i, j)$: for all vertices v s.t. $\ell(v) = i$, we set $\ell(v) = j$.

The *generated graph* is the one obtained from the k-expression by deleting all the labels. The clique-width of a graph G, denoted by $cw(G)$, is the least k such that it is the graph generated by some k-expression.

It is useful to see a k-expression as a rooted tree. Namely, the leaves of this tree are labeled by the operations $i(v)$ for vertex-creation. The internal nodes are labeled by the other operations, with the degree of each such node being the arity of the corresponding operation: 1 for the join operations and the relabeling operations, and 2 for the disjoint union operations. See Fig. 1 for an example. We call this tree representation the *parse tree* of the k-expression.

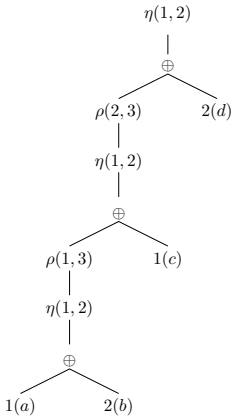

Figure 1 The parse tree of some 3-expression of P_4.

The size of a k-expression is its number of operations (= number of nodes in its parse tree). Throughout the remainder of the paper, we assume each given k-expression for a graph to be of linear size $\mathcal{O}(n + m)$. Note that it is always the case if we restrict ourselves to a subclass where a k-expression can be computed in linear time, that is anyway the relevant case for which our results in this paper could be applied. More generally, any k-expression can be transformed into an equivalent k-expression of size $\mathcal{O}(n + m)$ [39].

2.2 Tree-width

A *tree decomposition* $(T, \mathcal{X})$ of $G = (V, E)$ is a pair consisting of a tree T and of a family $\mathcal{X} = (X_t)_{t \in V(T)}$ of subsets of V indexed by the nodes of T and satisfying:

- $\bigcup_{t \in V(T)} X_t = V$;
- for any edge $e = \{u, v\} \in E$, there exists $t \in V(T)$ such that $u, v \in X_t$;
- for any $v \in V$, the set of nodes $\{t \in V(T) \mid v \in X_t\}$ induces a subtree T_v of T.

The sets X_t are called *the bags* of the decomposition. The *width* of a tree decomposition is the size of a largest bag minus one. Finally, the *tree-width* of a graph G, denoted by $\mathsf{tw}(G)$, is the least possible width over its tree decompositions. We only use tree-width in Sec. 5.

3 Courcelle's theorem

We refer to [14] for a thorough treatment of graph logics and their algorithmic applications. Recall that in *monadic second-order logic* (for short, MSO logic), we are given first-order variables x (written in lower-case), and set variables X (written in upper-case). We allow atomic formulas of the form $x \in X$, expressing the membership of x to a set X. The *counting* MSO (for short, $CMSO$) further allows atomic formulas of the form $Card_{p,q}(X)$, expressing that $|X| \equiv p \pmod q$ and sometimes called *counting predicates*. – We here assume p and q to be fixed constants, but it should be noted that in the general case, the complexity of $CMSO$ model checking also depends on the values of p and q. – The $CMSO$ logic is stronger than MSO logic: for instance, there is no MSO formula expressing that a set has even cardinality [14]. In Sec. 4, we will need the counting predicates of $CMSO$ logic in order to express the Tutte-Berge formula. Before that, we need to define $CMSO$ *optimization functions*.

Let φ be some $CMSO$ formula with $r + s$ free variables, and let $a_1, a_2, \ldots, a_s$ be fixed integers — possibly, null or negative. Let $Z_1, Z_2, \ldots, Z_r$ be fixed subsets of the domain of the first-order variables. We define $\psi(Z_1, Z_2, \ldots, Z_r)$ as the minimum value $\sum_{i=1}^{s} a_i \cdot |X_i|$ amongst all subsets $X_1, X_2, \ldots, X_s$ such that $\varphi(Z_1, Z_2, \ldots, Z_r, X_1, X_2, \ldots, X_s)$ is true. Then, ψ is a $CMSO$ optimization function of arity r. We define the size of ψ as $|\psi| = r + s$ (*a.k.a.*, as the arity of the underlying $CMSO$ formula φ).

To a c-labelled graph $G = (V, E, \ell)$, we can associate the *relational structure* $\langle V, \{inc, lab_1, lab_2, \ldots, lab_c\}\rangle$ where the vertex-set V is the domain of first-order variables, the binary operator $inc : V \times V \to \{0, 1\}$ asserts whether two vertices are adjacent in G and, for each i, $lab_i : V \to \{0, 1\}$ asserts whether the label of a given vertex equals i. The $(C)MSO_1$ logic on graphs is the above $(C)MSO$ logic restricted to such structures. We define $CMSO_1$ optimization functions in the exact same way. There is a more general $(C)MSO_2$ logic, where we also allow variables to represent edges, but it is not discussed here. Finally, define the underlying graph of G as the graph obtained from G by removing all its labels.

The following result was proved by Courcelle and Vanicat [17], but only for MSO_1 logic and for a more restricted type of optimization problems.

▶ **Theorem 3.** *Let ψ be a $CMSO_1$ optimization function on c-labelled graphs, for some fixed constant c, and of arity r. For every c-labelled graph G of clique-width at most k, if a k-expression is given for the underlying graph of G then, after a pre-processing in $\tilde{\mathcal{O}}_{k,|\psi|}(n + m)$ time, for every vertex-subsets $Z_1, Z_2, \ldots, Z_r$ of G, we can compute the value $\psi(Z_1, Z_2, \ldots, Z_r)$ in $\tilde{\mathcal{O}}_{k,|\psi|}\left(\sum_{j=1}^{r} |Z_j|\right)$ time.*

As a particular case of the above Theorem 3 (for $r = 0$), we retrieve the optimization version of Courcelle's theorem for bounded clique-width graphs (see [15]):

▶ **Theorem 4.** *Let φ be a $CMSO_1$ formula on c-labelled graphs, for some fixed constant c, and with s free variables. Let. also $a_1, a_2, \ldots, a_s$ be fixed integers. For every c-labelled graph G of clique-width at most k, if a k-expression is given for the underlying graph of G then, in $\tilde{\mathcal{O}}_{k,|\varphi|}(n+m)$ time, one can compute the minimum value $\sum_{i=1}^{s} a_i \cdot |X_i|$ amongst all vertex-subsets $X_1, X_2, \ldots, X_s$ such that $\varphi(X_1, X_2, \ldots, X_s)$ is true.*

Theorem 3 should not be considered as completely new. As it was stated above, it was first proved by Courcelle and Vanicat [17], but under more restricted conditions. Still, their proof also applies to this more general case. It could also be deduced from the heavy machinery from [14]. Our presentation marginally differs from these previous works, while it avoids using explicitly some logic concepts such as MSO transductions. Roughly, we rewrite a $CMSO_1$ formula on a c-labelled graph as a longer $CMSO$ formula on the parse tree of its clique-width expression. Then, we make this parse tree of logarithmic depth, using a modified centroid decomposition, updating the $CMSO$ formula along the way. We end up designing a dynamic programming procedure on this parse tree, using prior work of Doner [24] and Thatcher and Wright [67] on tree automata.

4 Algorithms: the general case

Our main result in this section (Theorem 1) is proved in Sec. 4.3. In Sec. 4.1 we first compute the matching number, a key step toward the final proof of Theorem 1. Sec. 4.2 is devoted to computing the Edmonds-Gallai decomposition, and it is a gentle introduction to the techniques we also use in Sec. 4.3.

4.1 Size of a maximum matching

We explain in this section how to compute the matching number of bounded clique-width graphs. For that, we need a classic result from Matching theory:

▶ **Lemma 5** (Tutte-Berge formula [6, 7])**.** *For any graph $G = (V, E)$, we have:*

$$\nu(G) = \min_{U \subseteq V} \frac{1}{2} \left(|V| + |U| - odd(G \setminus U) \right)$$

where $odd(G \setminus U)$ denotes the number of connected components of odd size of $G \setminus U$.

Our main insight below is that evaluating the Tutte-Berge formula can be written as a $CMSO_1$ optimization problem. We prove it next:

▶ **Theorem 6.** *For any graph $G = (V, E)$ of clique-width at most k, if a k-expression is given then, we can compute $\nu(G)$ in $\tilde{\mathcal{O}}_k(n+m)$ time.*

Proof. By Theorem 4, it suffices to prove that the Tutte-Berge formula (see Lemma 5) can be written as a $CMSO_1$ optimization problem. For that, let us first define $inc(x, y, U) := inc(x, y) \wedge x \notin U \wedge y \notin U$ in order to suppress all edges incident to a given set U. The following formula can be used to test whether two vertices are in the same connected component of $G \setminus U$, for a given set U: $connected(x, y, U) := \forall X ((x \in X \wedge y \notin X) \implies \exists x', y' (x' \in X \wedge y' \notin X \wedge inc(x', y', U)))$. Then, we can relate a vertex to its connected component of $G \setminus U$ as

follows: $comp(x, X, U) := \forall y(y \in X \iff connected(x, y, U))$. We are now ready to define our formula for computing $\nu(G)$. It has two free variables.

$$\begin{aligned} TutteBerge(U, W) :=& \forall x \in W(x \notin U \wedge \exists X(comp(x, X, U) \wedge Card_{1,2}(X))) \\ & \wedge \forall x, y \in W(x = y \vee \neg connected(x, y, U)). \end{aligned}$$

The first line ensures that every vertex of W is in an odd component of $G \setminus U$. The second line ensures that two distinct vertices of W are in different components of $G \setminus U$. If we set $a_1 = 1, a_2 = -1$, the objective becomes to minimize $|U| - |W|$. Therefore, we get as solution $\delta = \min_{U \subseteq V}(|U| - odd(G \setminus U))$. By Lemma 5, we have $\nu(G) = \frac{1}{2}(n + \delta)$. ◀

This above Theorem 6 is the cornerstone of all the remainder of Sec. 4.

4.2 Edmonds-Gallai decomposition

We continue with a known structural result about maximum matchings in a graph. Recall that a graph is hypomatchable if the removal of any one vertex results in a graph with a perfect matching.

▶ **Theorem 7** (Edmonds-Gallai [30, 42, 43]). *Let $G = (V, E)$ be a graph, and let $A \subseteq V$ be the set of all vertices v so that there is a maximum matching of G that does not cover v. Set $B = N_G(A)$ and $C = V \setminus (A \cup B)$. Then:*

- *Every odd component H of $G \setminus B$ is hypomatchable and it has $V(H) \subseteq A$;*
- *Every even component H of $G \setminus B$ has a perfect matching and it has $V(H) \subseteq C$;*
- *For every $X \subseteq B$, the set $N(X)$ contains vertices in $> |X|$ odd components of $G \setminus B$.*

The partition (A, B, C) is called the Edmonds-Gallai decomposition *of G.*

In [10], the author proposes a randomized $\mathcal{O}(n^\omega)$-time algorithm for computing the Edmonds-Gallai decomposition of an n-vertex graph, where $\omega < 2.37286$ [2] denotes the exponent of square matrix multiplication. We improve this result to deterministic almost linear-time for all classes of bounded clique-width graphs (under the standard assumption in the field that a corresponding clique-width expression is given in the input):

▶ **Theorem 8.** *For any graph $G = (V, E)$ of clique-width at most k, if a k-expression is given then, we can compute its Edmonds-Gallai decomposition in $\tilde{\mathcal{O}}_k(n + m)$ time.*

Proof. If we are given the set A of all vertices left exposed by at least one maximum matching then, by Theorem 7, the sets B and C can be computed in additional $\mathcal{O}(n + m)$ time. Recall (see Theorem 6) that there exists a $CMSO_1$ formula $TutteBerge(U, W)$ to express that all vertices of W are in pairwise different odd components of $G \setminus U$. Let $EdmondsGallai(X, U, W) := TutteBerge(U \cup X, W)$. It is also a $CMSO_1$ formula since the union of two subsets can be easily expressed in MSO [14]. Then, for any X, let $\psi(X)$ be the problem of minimizing $|U| - |W|$ among all the subsets U, W such that $EdmondsGallai(X, U, W)$ is true. Observe that ψ is a $CMSO_1$ optimization function. We apply Theorem 3 to ψ. Then, we claim that $v \in A$ if and only if $\psi(\{v\}) = 2\nu(G) + 1 - n$. Indeed, by construction we have $\psi(\{v\}) = \min_{U \subseteq V \setminus \{v\}}(|U| - odd(G \setminus (\{v\} \cup U)))$, and therefore by Lemma 5, $\nu(G \setminus \{v\}) = \frac{1}{2}(n - 1 + \psi(\{v\}))$. Then:

$$\begin{aligned} v \in A \iff & \nu(G) = \nu(G \setminus \{v\}) \iff \nu(G) = \frac{1}{2}(n - 1 + \psi(\{v\})) \\ \iff & 2\nu(G) = n - 1 + \psi(\{v\}) \iff \psi(\{v\}) = 2\nu(G) + 1 - n. \end{aligned}$$

Computing $\nu(G)$ can be done in $\tilde{\mathcal{O}}_k(n+m)$ time (Theorem 6). Computing $\psi(\{v\})$ takes $\tilde{\mathcal{O}}_k(1)$ time per vertex v up to an $\tilde{\mathcal{O}}_k(n+m)$-time pre-processing (Theorem 3). As a result, we can compute the set A, and so, the Edmonds-Gallai decomposition, in $\tilde{\mathcal{O}}_k(n+m)$ time. ◀

4.3 Computation of a maximum matching

Let us first recall our main result in this section:

▶ **Theorem 1.** *Given a graph G and a corresponding k-expression, one can compute a maximum matching for G in deterministic $\tilde{\mathcal{O}}_k(n+m)$ time.*

Let us sketch our strategy to prove this above result. Given a graph $G=(V,E)$, we first recall that an edge-cut is, for a given subset A, the set of all edges between A and $V \setminus A$. It is balanced if we further have $\max\{|A|, |V \setminus A|\} \leq 2n/3$. Roughly, we compute a balanced edge-cut for G, we compute a subset of edges of the cut to be included in some maximum matching of G, then we recurse on subgraphs of $G[A]$ and $G[V \setminus A]$ separately.

The computation of a balanced edge-cut in $\tilde{\mathcal{O}}(k \cdot (n+m))$ time follows from prior works [19, 27] and is omitted here due to lack of space. An important property for this cut is that it can be edge-partitioned into at most k joins. We handle each join separately. For that, both Lemma 9 and Lemma 10 below apply Theorem 3 (distributed Courcelle's theorem).

▶ **Lemma 9.** *Let X, Y be the two sides of a join in a graph $G=(V,E)$, where $|X| \leq |Y|$ and $cw(G) \leq k$. If a k-expression is given then, in $\tilde{\mathcal{O}}_k(n+m)$ time, we can compute an inclusion-wise minimal subset $X' \subseteq X$ such that, in some maximum matching of G:*
1. *every vertex of X' is matched to some vertex of Y;*
2. *no vertex of $X \setminus X'$ is matched to a vertex of Y.*

▶ **Lemma 10.** *Let X, Y be the two sides of a join in a graph $G=(V,E)$, where $|X| \leq |Y|$ and $cw(G) \leq k$. We are given a subset $X' \subseteq X$ as stated in Lemma 9. If a k-expression is given then, in $\tilde{\mathcal{O}}_k(n+m)$ time, we can compute the intersection of the edges of the join with some maximum matching of G.*

Finally, once we computed from a join a subset of edges to be added into a maximum matching, all other edges of the join can be removed from the graph. We must also remove all the end-vertices of the edges included into the matching. Doing so, we need the following two lemmas in order to update the k-expression of the graph considered.

▶ **Lemma 11.** *Let $G=(V,E)$ be a graph, let (U,W) be a cut of G, and let $U' \subseteq U$. If $cw(G) \leq k$ then the graph H, obtained from G by removing all edges between U' and $W' := N(U') \cap W$, has clique-width at most $3k$. Furthermore, we can compute a $3k$-expression of H from a k-expression of G in $\mathcal{O}(n+m)$ time.*

▶ **Lemma 12.** *Let $G=(V,E)$ be a graph and let $H=(X,E_X)$ be an induced subgraph of G. If a k-expression of G is given then, in $\mathcal{O}(k \cdot (n+m))$ time, we can compute a k-expression of H of size $\mathcal{O}(|X|+|E_X|)$.*

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

Sketch Proof of Theorem 1. We compute some balanced cut (U,W) and a partition $U_1, U_2, \ldots, U_k$ of U such that the edges of the cut are partitioned into k joins (one of them having possibly no edge) with respective sides U_i and $W_i = N(U_i) \cap W$ for every i. Details are omitted due to lack of space. Since we are given the U_i's, all the corresponding sides W_i can be computed in total $\mathcal{O}(n+m)$ time. We consider these k joins sequentially, from $i=1$

to $i = k$. At each step i, we are given a subgraph G_i of G such that $cw(G_i) \leq 3k$ and a corresponding $3k$-expression is given (initially, $G_1 := G$). We apply Lemmas 9 and 10 in order to compute the intersection of a maximum matching of G_i with the join with sides U_i, W_i. It takes $\tilde{\mathcal{O}}_k(n+m)$ time. Denote $F_i \subseteq U_i \times W_i$ the set of edges in this intersection, and let $V(F_i)$ be the set of vertices incident to an edge of F_i. We obtain G_{i+1} from G_i by removing the vertices in $V(F_i)$ and all remaining edges between $U_i \setminus V(F_i)$ and $W_i \setminus V(F_i)$.

Let $M_i := \bigcup_{j=1}^{i} F_j$ be the matching constructed so far, and let $V(M_i)$ be the set of vertices incident to it. Let. also $X_i := \bigcup_{j=1}^{i} U_j$. By induction, the union of M_i with a maximum matching of G_{i+1} is a maximum matching of G. Also by induction, G_{i+1} is obtained from $G \setminus V(M_i)$ by removing all edges between $X_i \setminus V(M_i)$ and $W \setminus V(M_i)$. Therefore, we can apply Lemma 12 (for $X = V \setminus V(M_i)$) then Lemma 11 (for $U' = X_i \setminus V(M_i)$) to compute a $3k$-expression of G_{i+1}. It takes $\mathcal{O}(k \cdot (n+m))$ time. – Note that since we always apply Lemma 11 to G, and not to the G_i's, there is no blow-up of the clique-width value, *i.e.*, the clique-width of any G_i is at most $3k$. –

Since all k joins got removed, we are left with computing a maximum matching in $G[U \setminus V(M_k)]$ and in $G[W \setminus V(M_k)]$ respectively. Apply Lemma 12 to compute k-expressions for both subgraphs, then call our above algorithm recursively in order to compute a maximum matching. Since the cut is balanced, the recursive depth is in $\mathcal{O}(\log n)$. ◀

5 Algorithms: the bipartite case

We present in this section an alternative to Theorem 1, with polynomial dependency on the clique-width, but only for bipartite graphs (see Theorem 2). The structure of bipartite graphs of bounded clique-width can be quite complex. For instance, let $G = (V, E)$ be any graph and let $V' = \{v' \mid v \in V\}$ be a disjoint copy of V. We can define the bipartite graph $B_G = (V \cup V', \{u'v \mid uv \in E\})$ and, if G has clique-width at most k, then B_G has clique-width at most $2k$. Some classes of monogenic bipartite graphs are also known to have bounded clique-width [21].

5.1 Reduction to Maximum b-matching

Let $G = (V, E)$ be a graph and let $b : V \to \mathbb{N}$ assign a non negative capacity to each vertex. We say that $x : E \to \mathbb{N}$ is a b-matching if we have $\sum_{u \in N_G(v)} x_{uv} \leq b(v)$ for each vertex v. Observe that a matching is a b-matching for the trivial function $b(v) = 1$ for each $v \in V$. The cardinality of a b-matching is defined as $\|x\|_1 = \sum_{e \in E} x_e$. We denote by $\nu(G, b)$ the cardinality of a maximum b-matching in G. Let also $\|b\|_1 = \sum_{v \in V} b(v)$ be the sum of all the vertex capacities. More generally, for every vertex-subset S, let $b(S) = \sum_{v \in S} b(v)$. Given a b-matching x, and a vertex v, let also $d_x(v) := \sum_{u \in N_G(v)} x_{uv} \leq b(v)$.

We start with the following reduction rule:

▶ **Lemma 13.** *Let (G, b) be some instance of* Maximum b-Matching*, and let U and W be disjoint vertex-subsets such that there is a join between U and W. Consider the new instance (G', b') obtained from (G, b) by removing all edges between U and W, adding two new vertices $u, w \notin V(G)$ and edges $\{uw\} \cup \{uv \mid v \in U\} \cup \{wv' \mid v' \in W\}$, and finally setting $b'(u) = b'(w) = \min\{b(U), b(W)\}$. Then, we have $\nu(G', b') = \nu(G, b) + \min\{b(U), b(W)\}$.*

Moreover, if G is bipartite, then so is G'.

By repeatedly applying Lemma 13 to some special balanced edge-cuts, we obtain:

Figure 2 Transformation of Lemma 13.

▶ **Proposition 14.** *There is an $\mathcal{O}(k\cdot(n+m)\log n)$-time reduction from* Maximum Matching *on graphs with clique-width at most k (if a k-expression is known) to* Maximum b-Matching *on graphs with tree-width $\mathcal{O}(k\log n)$. For the resulting instance (H,b), the algorithm also outputs a corresponding tree decomposition. Furthermore, $|V(H)| \leq \|b\|_1 \leq \mathcal{O}(\min\{n+m, n\log n\})$, and if G is bipartite, then so is H.*

5.2 Reduction to Linear Programming

The Maximum b-Matching problem is a classic example of an integer linear program. It is well-known that for the special case of Maximum Matching within *bipartite* graphs, we can drop the condition for all variables to be integers, thus reducing the computation of the matching number to the solving of a linear program [55]. Because of Tutte's quasi-polynomial reduction from Maximum b-Matching to Maximum Matching [69], this is also true for Maximum b-Matching within bipartite graphs. Very recently, Dong et. al. [25] answered an open question from Fomin et. al. [36] about bounded tree-width linear programs. We restate below their main result:

▶ **Theorem 15** ([25])**.** *Given a linear program $\max_{Mx=b,\ell\leq x\leq u} c^\intercal x$ [4], where $M\in\mathbb{R}^{d\times n}$ is a full-rank matrix with $d\leq n$, define the* **dual graph** *G_M to be the graph with vertex-set $\{1,2\ldots,d\}$, such that $ij\in E(G_M)$ if there is a column r such that $M_{i,r}\neq 0$ and $M_{j,r}\neq 0$. Suppose that a tree decomposition of G_M with width τ is given, and R is the diameter of the polytope, namely, for any $\ell\leq x\leq u$ with $Mx=b$, we have $\|x\|_2\leq R$. Then, for any $0<\varepsilon\leq 1$, we can find $\ell\leq x\leq u$ such that*

$$c^\intercal x \geq \max_{Mx=b,\ell\leq x\leq u} c^\intercal x - \varepsilon\cdot\|c\|_2\cdot R \text{ and } \|Mx-b\|_2\leq\varepsilon\cdot(\|M\|_2\cdot R+\|b\|_2)$$

in expected time $\tilde{\mathcal{O}}(n\cdot\tau^2\log(1/\varepsilon))$.

Dong et. al. [25] referred to [66] and [53] for a detailed discussion about converting an approximate solution to an exact solution. We give a direct proof for Maximum b-Matching within bipartite graphs. For that, we combine a folklore reduction to Maximum Flow with a nice rounding technique by Madry [56].

▶ **Proposition 16.** *The* Maximum b-Matching *problem within bipartite graphs of tree-width at most τ can be solved in expected $\tilde{\mathcal{O}}(n\tau^2)$ time, if a corresponding tree decomposition is given in the input.*

[4] The result is stated in [25] for minimization problems. Since we only consider it here for Maximum b-Matching, we rather write it as a maximization problem.

Proof. Let $G = (V_0 \cup V_1, E)$ be a bipartite graph and let $b : V_0 \cup V_1 \to \mathbb{N}$. The incidence matrix of G is the $n \times m$ matrix M such that $M_{v,e} = 1$ if v is an end-vertex of edge e and $M_{v,e} = 0$ otherwise. Let also c be the all-one vector. To compute the cardinality of a maximum b-matching for G, it suffices to solve the linear program $\max_{Mx \leq b} c^\intercal x$. We slightly modify this above program so that we fit in the conditions of Theorem 15. First, if we let $\ell, u \in \mathbb{R}^m$ such that ℓ is all-zero and $u_{vv'} = \min\{b(v), b(v')\}$ for each edge $vv' \in E$, then we must now solve $\max_{Mx \leq b, \ell \leq x \leq u} c^\intercal x$. If we further add n new variables $(x_v)_{v \in V}$ such that $0 \leq x_v \leq b(v)$ for each $v \in V$, then we can replace all constraints by $x_v + \sum_{vv' \in E} x_{vv'} = b(v)$ for each vertex $v \in V$, thus getting a new linear program $\max_{M'x = b, \ell' \leq x \leq u'} c'^\intercal x$ to solve, where $M' \in \mathbb{R}^{n \times (m+n)}$.

Note that, since we constructed M' from M by adding n new columns with exactly one nonzero value each, we have $G_{M'} = G_M = G$. Furthermore, an easy upper bound on the diameter R of the polytope is $R \leq \|b\|_2 \leq \|b\|_1$. We also have $\|c'\|_2 = \sqrt{m}$ and $\|M'\|_2 \leq \sqrt{\sum_v (1 + d(v))^2} \leq n^{3/2}$.

Assume G to be given with a tree decomposition of width at most τ, and apply Theorem 15 to the above linear program with $\varepsilon = 1/(4 \cdot \|b\|_1 \cdot n^2)$. We denote by x its output. Set all variables $x_v, v \in V$ to 0. Then, for all vertices v such that $\sum_{vv' \in E} x_{v'v} > b(v)$, we decrease the variables $x_{v'v}$ of incident edges until we reach equality. In doing so, we obtain a fractional b-matching y. By construction:

$$\begin{aligned}\|x\|_1 - \|y\|_1 &\leq \sum_v \max\{0, (M'x)_v - b(v)\} \leq \|M'x - b\|_1 \leq \sqrt{n} \cdot \|M'x - b\|_2 \\ &\leq \varepsilon\sqrt{n} \cdot (\|M'\|_2 \cdot R + \|b\|_2) \leq \varepsilon\sqrt{n} \cdot \|b\|_1 \cdot (n^{3/2} + 1) \leq 2\varepsilon n^2 \cdot \|b\|_1 \leq 1/2.\end{aligned}$$

We now construct a network D from G by adding two new vertices s and t, an arc (s, u) for every $u \in V_0$, an arc (v, t) for every $v \in V_1$, and finally by orienting all the edges of G from V_0 to V_1. The capacities of the arcs are defined as follows: $\kappa(s, v) = b(v)$ for every $v \in V_0$; $\kappa(v', t) = b(v')$ for every $v' \in V_1$; $\kappa(v, v') = \min\{b(v), b(v')\}$ for every $vv' \in E$ such that $v \in V_0, v' \in V_1$. Then, we construct a fractional st-flow as follows: $f_y(s, v) = \sum_{vv' \in E} y_{vv'}$ for every $v \in V_0$; $f_y(v', t) = \sum_{vv' \in E} y_{vv'}$ for every $v' \in V_1$; and $f_y(v, v') = y_{vv'}$ for every $vv' \in E$ such that $v \in V_0, v' \in V_1$. Note that the value of this flow is exactly $\|y\|_1$. Let f'_y be an integral st-flow of value $\lfloor \|y\|_1 \rfloor$. It can be constructed from f_y in $\tilde{\mathcal{O}}(m) = \tilde{\mathcal{O}}(\tau n)$ time [56, Corollary 3.4]. There is a one-to-one mapping between b-matchings in G and integral st-flows in D. In particular, the maximum value of a st-flow is exactly $\nu(G, b)$. Furthermore,

$$\|y\|_1 \geq \|x\|_1 - 1/2 \geq \nu(G, b) - \varepsilon \cdot \|c\|_2 \cdot R - 1/2 \geq \nu(G, b) - 1.$$

Therefore, we can transform f'_y into a maximum st-flow f_z by computing at most one augmenting path in the residual graph $D_{f'_y}$. It can be done in $\mathcal{O}(m) = \mathcal{O}(\tau n)$ time. The resulting b-matching is maximum: for every $vv' \in E$ with $v \in v_0, v' \in V_1$, we set $z_{vv'} = f_z(v, v')$. ◀

We are finally ready to prove our second main result in this paper:

▶ **Theorem 2.** *Given a bipartite graph G and a corresponding k-expression, one can compute a maximum matching for G in randomized $\tilde{\mathcal{O}}(k^2 \cdot (n + m))$ time.*

Proof. The result follows from the combination of Proposition 14 with Proposition 16. ◀

6 Conclusion

Our first main result in the paper is a quasi linear-time algorithm for computing a maximum matching within the graphs of bounded clique-width. We left open whether an $\tilde{\mathcal{O}}(k^c \cdot (n+m))$-time algorithm exists within the graphs of clique-width at most k, for some constant c. Our second main result is to prove the existence of such an algorithm for the bipartite graphs of clique-width at most k. For that, we reduce the Maximum Matching problem on bounded clique-width (bipartite) graphs to the Maximum b-Matching problem on bounded tree-width (bipartite) graphs. We left open the complexity of Maximum b-Matching within bounded tree-width graphs in general. Furthermore, we observe that an $\tilde{\mathcal{O}}(k^c \cdot (n+m))$-time algorithm for this problem within the graphs of tree-width at most k would imply a similar algorithm for Maximum Matching within graphs of clique-width at most k.

References

1 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.

2 J. Alman and V. Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

3 N. Anari and V. Vazirani. Matching Is as Easy as the Decision Problem, in the NC Model. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

4 R.P. Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24(3):153–157, 1987.

5 M. Bentert, T. Fluschnik, A. Nichterlein, and R. Niedermeier. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences*, 103:61–77, 2019.

6 C. Berge. Sur le couplage maximum dun graphe. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L'ACADEMIE DES SCIENCES*, 247(3):258–259, 1958.

7 John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer-Verlag London, 2008.

8 M.L. Carmosino, J. Gao, R. Impagliazzo, I. Mihajlin, R. Paturi, and S. Schneider. Non-deterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 261–270, 2016.

9 M. Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In *ISAAC*, pages 146–155. Springer, 1996.

10 J. Cheriyan. Randomized $\tilde{\mathcal{O}}(M(|V|))$ Algorithms for Problems in Matching Theory. *SIAM Journal on Computing*, 26(6):1635–1655, 1997.

11 V. Cohen-Addad, M. Habib, and F. de Montgolfier. Algorithmic aspects of switch cographs. *Discrete Applied Mathematics*, 200:23–42, 2016.

12 Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005. doi:10.1137/S0097539701385351.

13 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.

14 B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

15 B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

16 B. Courcelle and A. Twigg. Constrained-path labellings on graphs of bounded clique-width. *Theory of Computing Systems*, 47(2):531–567, 2010.

17 B. Courcelle and R. Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Applied Mathematics*, 131(1):129–150, 2003.

18 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

19 Bruno Courcelle, Pinar Heggernes, Daniel Meister, Charis Papadopoulos, and Udi Rotics. A characterisation of clique-width through nested partitions. 187:70–81, 2015. doi:10.1016/j.dam.2015.02.016.

20 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. 101(1):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

21 K.K. Dabrowski and D. Paulusma. Classifying the clique-width of H-free bipartite graphs. *Discrete Applied Mathematics*, 200:43–51, 2016.

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

23 Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2010. *4th* edition. doi:10.1007/978-3-662-53622-3.

24 J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4(5):406–451, 1970.

25 S. Dong, Y.T. 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.

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

27 G. Ducoffe. Optimal diameter computation within bounded clique-width graphs. Technical Report 2011.08448, arXiv, 2020.

28 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.

29 G. Ducoffe and A. Popa. The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes. *Discrete Applied Mathematics*, 291:201–222, 2021.

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

31 Wolfgang Espelage, Frank Gurski, and Egon Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. volume 1, pages 117–128. Springer, 2001. doi:10.1007/3-540-45477-2_12.

32 T. Fluschnik, G.B. Mertzios, and A. Nichterlein. Kernelization lower bounds for finding constant-size subgraphs. In *Conference on Computability in Europe*, pages 183–193. Springer, 2018.

33 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM Journal on Computing*, 39(5):1941–1956, 2010. doi:10.1137/080742270.

34 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014. doi:10.1137/130910932.

35 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Clique-width III: Hamiltonian Cycle and the Odd Case of Graph Coloring. 15(1):9, 2019. doi:10.1145/3280824.

36 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michal Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. 14(3):34:1–34:45, 2018. doi:10.1145/3186898.

37 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.

38 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.

39 Martin Fürer. A natural generalization of bounded tree-width and bounded clique-width. In *Latin American Symposium on Theoretical Informatics*, pages 72–83. Springer, 2014.

40 H.N. Gabow. Data structures for weighted matching and extensions to b-matching and f-factors. *ACM Transactions on Algorithms (TALG)*, 14(3):1–80, 2018.

41 H.N. Gabow and P. Sankowski. Algorithms for Weighted Matching Generalizations I: Bipartite Graphs, b-matching, and Unweighted f-factors. *SIAM Journal on Computing*, 50(2):440–486, 2021.

42 T. Gallai. Kritische graphen II. *Magyar Tud. Akad. Mat. Kutato Int. Kozl.*, 8:373–395, 1963.

43 T. Gallai. Maximale systeme unabhangiger kanten. *Magyar Tud. Akad. Mat. Kutato Int. Kozl.*, 9:401–413, 1964.

44 A. Gerards. Matching. *Handbooks in operations research and management science*, 7:135–224, 1995.

45 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science*, 689:67–95, 2017. doi:10.1016/j.tcs.2017.05.017.

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

47 Martin Charles Golumbic and Udi Rotics. On the clique-width of some perfect graph classes. 11(03):423–443, 2000. doi:10.1142/S0129054100000260.

48 Torben Hagerup, Jyrki Katajainen, Naomi Nishimura, and Prabhakar Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *Journal of Computer and System Sciences*, 57(3):366–375, 1998.

49 F. Hegerfeld and S. Kratsch. On Adaptive Algorithms for Maximum Matching. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 71:1–71:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

50 Yoichi Iwata, Tomoaki Ogasawara, and Naoto Ohsaka. On the power of tree-depth for fully polynomial FPT algorithms. volume 96, pages 41:1–41:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.41.

51 S. Kratsch and F. Nelles. Efficient and Adaptive Parameterized Algorithms on Modular Decompositions. pages 55:1–55:15, 2018. doi:10.4230/LIPIcs.ESA.2018.55.

52 S. Kratsch and F. Nelles. Efficient Parameterized Algorithms for Computing All-Pairs Shortest Paths. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, volume 154, pages 38:1–38:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

53 Y. T. Lee and A. Sidford. Path Finding I: Solving Linear Programs with\˜ O (sqrt (rank)) Linear System Solves. Technical Report 1312.6677, arXiv, 2013.

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

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

56 A. Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 253–262. IEEE, 2013.

57 Johann A. Makowsky and Udi Rotics. On the clique-width of graphs with few P_4's. 10(03):329–348, 1999. doi:10.1142/S0129054199000241.

58 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.

59 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. The power of linear-time data reduction for maximum matching. volume 83, pages 46:1–46:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.MFCS.2017.46.

60 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.

61 A. Moitra and R. Johnson. A parallel algorithm for maximum matching on interval graphs. In *ICPP*, 1989.

62 S. Oum and P. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

63 M.W. Padberg and M.R. Rao. The Russian method for linear inequalities III: Bounded integer programming. Technical Report 78, INRIA, 1981.

64 W.R. Pulleyblank. *Faces of Matching Polyhedra*. PhD thesis, Univ. of Waterloo, Dept. Combinatorics and Optimization, 1973.

65 Michaël Rao. Solving some NP-complete problems using split decomposition. 156(14):2768–2780, 2008. doi:10.1016/j.dam.2007.11.013.

66 J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical programming*, 40(1):59–93, 1988.

67 J.W. Thatcher and J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2(1):57–81, 1968.

68 N. Trinajstić, D.J. Klein, and M. Randić. On some solved and unsolved problems of chemical graph theory. *International Journal of Quantum Chemistry*, 30(S20):699–742, 1986.

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

70 W.T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947.

71 W.T. Tutte. Antisymmetrical digraphs. *Canadian Journal of Mathematics*, 19:1101–1117, 1967.

72 V. Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018.

73 V. Vassilevska Williams and R.R. Williams. Subcubic equivalences between path, matrix, and triangle problems. *Journal of the ACM (JACM)*, 65(5):1–38, 2018.

74 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.

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

76 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.