On Adaptive Algorithms for Maximum Matching

Falko Hegerfeld
Humboldt-Universität zu Berlin, Germany
hegerfeld@informatik.hu-berlin.de

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany
kratsch@informatik.hu-berlin.de

Abstract

In the fundamental Maximum Matching problem the task is to find a maximum cardinality set of pairwise disjoint edges in a given undirected graph. The fastest algorithm for this problem, due to Micali and Vazirani, runs in time $\mathcal{O}(\sqrt{n}m)$ and stands unbeaten since 1980. It is complemented by faster, often linear-time, algorithms for various special graph classes. Moreover, there are fast parameterized algorithms, e.g., time $\mathcal{O}(km \log n)$ relative to tree-width k, which outperform $\mathcal{O}(\sqrt{n}m)$ when the parameter is sufficiently small.

We show that the Micali-Vazirani algorithm, and in fact any algorithm following the phase framework of Hopcroft and Karp, is adaptive to beneficial input structure. We exhibit several graph classes for which such algorithms run in linear time $\mathcal{O}(n+m)$. More strongly, we show that they run in time $\mathcal{O}(\sqrt{k}m)$ for graphs that are k vertex deletions away from any of several such classes, without explicitly computing an optimal or approximate deletion set; before, most such bounds were at least $\Omega(km)$. Thus, any phase-based matching algorithm with linear-time phases obliviously interpolates between linear time for $k = \mathcal{O}(1)$ and the worst case of $\mathcal{O}(\sqrt{n}m)$ when $k = \Theta(n)$. We complement our findings by proving that the phase framework by itself still allows $\Omega(\sqrt{n})$ phases, and hence time $\Omega(\sqrt{n}m)$, even on paths, cographs, and bipartite chain graphs.

2012 ACM Subject Classification Mathematics of computing → Matchings and factors; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Matchings, Adaptive Analysis, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.71

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.11244`.

1 Introduction

The objective in the fundamental Maximum Matching problem is to find a set of disjoint edges of maximum cardinality in a given undirected graph $G = (V, E)$. Maximum Matching has been heavily studied and was the first problem for which a polynomial-time algorithm has explicitly been established [17]. Several algorithms [8, 23, 27, 39] achieve the best known running time of $\mathcal{O}(\sqrt{n}m)$ for graphs with n vertices and m edges, starting with the algorithm of Micali and Vazirani [39] in 1980. Since then, the time of $\mathcal{O}(\sqrt{n}m)$ remains unbeaten.

This state-of-the-art has motivated extensive research into faster algorithms for Maximum Matching on special inputs: Extensive effort went into beating the worst case running time of $\mathcal{O}(\sqrt{n}m)$ for Maximum Matching on special graph classes, resulting in a large number of publications [14, 21, 22, 24, 26, 29, 34, 35], often even obtaining linear-time algorithms [10, 12, 38, 43, 45]. Similarly, there is a great variety of algorithms whose running time depends on n and m but also on some structural parameter of the input graph, like its tree-width, its genus, or its vertex-deletion distance to a certain graph class (summarized in Table 2). As an example, one can solve Maximum Matching in time $\mathcal{O}(k(n+m))$ when

© Falko Hegerfeld and Stefan Kratsch;
licensed under Creative Commons License CC-BY
46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 71; pp. 71:1–71:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 The second column shows the running times of dedicated algorithms for MAXIMUM MATCHING on restricted inputs, which follow as special cases of previous work (see Table 2). Columns three and four show our results for algorithms employing the phase framework with linear-time phases. Only vertex cover number and matching number had known time $\mathcal{O}(\sqrt{k}m)$, others had $\Omega(km)$ prior to our work. The parameters mw and md refer to modular-width and modular-depth.

	Dedicated algorithm	Phase framework algorithms	
Parameter / Graph class	Parameter s	Parameter s	Dist. k to parameter s
Vertex cover number	$\mathcal{O}(\sqrt{s}m)$	$\mathcal{O}(\sqrt{s}m)$	n.a.
Star forest	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(\sqrt{k}m)$
Bounded tree-depth	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(\sqrt{k}m)$
Cluster graph	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(\sqrt{k}m)$
Minimum degree $n-s$	$\mathcal{O}(sn^2 \log n)$	$\mathcal{O}(sm)$	$\mathcal{O}(\sqrt{k}sm)$
Independence number	none	$\mathcal{O}(sm)$	$\mathcal{O}(\sqrt{k}s^2m)$
Neighborhood diversity	$\mathcal{O}((s^2 \log s)n + m)$	$\mathcal{O}(sm)$	$\mathcal{O}(\sqrt{k}sm)$
Parameter mw and md	$\mathcal{O}((\mathrm{mw}^2 \log \mathrm{mw})n + m)$	$\mathcal{O}((c\,\mathrm{mw})^{\mathrm{md}}m)$	$\mathcal{O}(\sqrt{k}(c\,\mathrm{mw})^{\mathrm{md}}m)$

the input graph G is given together with a set S of k vertices such that $G - S$ belongs to a class $\mathcal{C}$ in which MAXIMUM MATCHING can be solved in linear time (cf. [38]): It suffices to solve the problem on $G - S$ in linear time and to then apply at most k augmentation steps to account for vertices in S; each such step can be implemented in linear time.

A caveat of this great number of different algorithms for special cases is that we may have to first find the relevant structure, e.g., a set S so that $G - S$ belongs to a certain graph class $\mathcal{C}$, and to then decide which algorithm to apply. In some cases, finding the relevant optimal structure is NP-hard and using approximate structure may lead to increase in running time. Moreover, except for time $\mathcal{O}(\sqrt{k}m)$ relative to vertex cover number k or maximum matching size k, which can be seen to follow from the general analysis of Hopcroft and Karp [29], the previously known time bounds improve on $\mathcal{O}(\sqrt{n}m)$ only if $k = \mathcal{O}(\sqrt{n})$, at best.

Our results. We approach the MAXIMUM MATCHING problem from the perspective of *adaptive analysis (of algorithms)*. Rather than developing further specialized algorithms for special classes of inputs, we prove that a single algorithm actually achieves the best time bounds relative to several graph classes and parameters; in particular, several new or improved time bounds are obtained. Moreover, that algorithm is known since 1980, namely it is the Micali-Vazirani-algorithm [39], and it is oblivious to the actual structure and parameter values. In fact, our analysis does not depend on overly specific aspects of that algorithm and, rather, applies to any algorithm for MAXIMUM MATCHING that follows the "phase framework" established by Hopcroft and Karp [29] for BIPARTITE MATCHING and that implements each phase in linear time, e.g., the algorithms by Blum [8] and Goldberg and Karzanov [27]. In this framework, each phase is dedicated to finding a disjoint and maximal packing of shortest augmenting paths, and it can be shown that $\mathcal{O}(\sqrt{n})$ phases always suffice (cf. [29]).

We show that algorithms following the phase framework adapt to beneficial structure in the form of inputs from special graph classes or inputs that are few vertex deletions away from such a class, without running a recognition algorithm or computing the deletion distance (i.e., they are obliviously adaptive). Concretely, we show that any such algorithm solves MAXIMUM MATCHING in linear time on several graph classes such as cluster graphs or graphs of bounded neighborhood diversity. Moreover, for many such classes we also show that any such algorithm takes time $\mathcal{O}(\sqrt{k}m)$ on graphs that are k vertex deletions away from the class, without explicitly computing such a set of deletions (or even knowing the class in question). Furthermore, this running time interpolates between the worst-case time $\mathcal{O}(\sqrt{n}m)$ for $k \in \Theta(n)$ and linear time for $k \in \Theta(1)$, hence remaining competitive even in the absence of beneficial input structure. Except for the matching number and the vertex cover number, time bounds of the form $\mathcal{O}(\sqrt{k}m)$ are new, even for dedicated algorithms. Besides that, we improve upon the algorithm by Yuster [46] for the special case of minimum degree $n - s$ and we improve upon the algorithm by Kratsch and Nelles [33] for the special case of bounded neighborhood diversity. Our positive results are summarized in Table 1.

We complement our findings by exhibiting several graph classes on which the phase framework still allows the worst-case of $\Theta(\sqrt{n})$ phases, and hence $\Theta(\sqrt{n}m)$ time. We prove this for paths, trivially perfect graphs, which are a subclass of cographs, and bipartite chain graphs (and hence their superclasses). This, of course, does not contradict the existence of dedicated linear-time algorithms nor the possibility of tweaking a phase-based algorithm to avoid the obstructions. Nevertheless, these results do rule out the possibility of proving adaptiveness of arbitrary phase-based algorithms, and they showcase obstructions that need to be handled to obtain more general adaptive algorithms for Maximum Matching.

Related work. Our work fits into the recent program of "FPT in P"[1] or efficient parameterized algorithms, initiated independently by Abboud et al. [1] and Giannopoulou et al. [25]. This program seeks to apply the framework of parameterized complexity to tractable problems to obtain provable running times relative to certain parameters that outperform the fastest known algorithms or (conditional) lower bounds obtained in the fine-grained analysis program (see, e.g., [2, 9, 41]). In particular, Mertzios et al. [38] have suggested Maximum Matching as the "drosophila" of FPT in P, i.e., as a central subject of study, similar to the role that Vertex Cover plays in parameterized complexity. Already, there is a large number of publications on parameterized algorithms for Maximum Matching [11, 15, 16, 20, 33, 36], apart from large interest in FPT in P in general [1, 6, 19, 30, 31, 32]. There has also been interest in linear-time preprocessing, which, relative to some parameter k, reduces the problem to solving an instance of size $f(k)$ and leads to time bounds of the form $\mathcal{O}(n+m+g(k))$ [37].

Adaptive analysis of algorithms has been most successful in the context of sorting and searching [3, 4, 13, 18]. We are not aware of prior (oblivious) adaptive analysis of established algorithms for the Maximum Matching problem but two works have designed dedicated adaptive algorithms relative to tree-depth [31] and modular-width [33].

Bast et al. [5] analyzed the Micali-Vazirani algorithm for random graphs, obtaining a running time of $\mathcal{O}(m \log n)$ with high probability.

Organization. Some preliminaries on graphs and matchings are recalled in Section 2. Section 3 is dedicated to recalling the analysis of Hopcroft and Karp [29] and defining phase-based algorithms. In Section 4 we present the positive results, except for results related to neighborhood diversity and modular decomposition which can be found in the full version of the paper. The lower bounds for paths and trivially perfect graphs are given in Section 5; the lower bound for bipartite chain graphs is in the full version of the paper. We conclude in Section 6.

[1] An FPT-algorithm solves a given (usually NP-hard) problem in time $f(k)n^c$ where k is some problem-specific parameter and n is the input size.

Table 2 Known parameterized algorithms for Maximum Matching, k denotes the corresponding parameter value and ω is the matrix multiplication constant.

Parameter	Running Time	Reference
Matching Number	$\mathcal{O}(\sqrt{k}m)$	[8, 27, 29, 39]
Vertex Cover Number	$\mathcal{O}(\sqrt{k}m)$ / $\mathcal{O}(k(n+m)$ / $\mathcal{O}(n+m+k^3)$	[29, 39] / [36] / [36]
Feedback Vertex Number	$\mathcal{O}(k(n+m))$ / $\mathcal{O}(kn+2^{\mathcal{O}(k)})$	[36] / [37]
Feedback Edge Number	$\mathcal{O}(k(n+m))$ / $\mathcal{O}(n+m+k^{1.5})$	[36] / [37]
$\Delta(G)-\delta(G)$	$\mathcal{O}(kn^2\log n)$	[46]
Tree-width	$\mathcal{O}(k^4 n\log n)$ / $\mathcal{O}(km\log n)$	[20] / [31]
Tree-depth	$\mathcal{O}(km)$	[31]
Modular-width	$\mathcal{O}(k^4 n+m)$ / $\mathcal{O}((k^2\log k)n+m)$	[11] / [33]
Split-width	$\mathcal{O}((k\log^2 k)(n+m)\log n)$	[15]
P_4-sparseness	$\mathcal{O}(k^4(n+m))$	[11]
Genus	$\mathcal{O}(f(k)n^{\omega/2})$	[47]
H-minor-free	$\mathcal{O}(f(H)n^{3\omega/(\omega+3)})$	[47]
Dist. to Cocomparability	$\mathcal{O}(k(n+m))$	[38]
Distance to Chain Graph	$\mathcal{O}(k(n+m))$ / $\mathcal{O}(n+m+k^3)$	[36] / [37]

2 Preliminaries

We mostly consider simple graphs, unless stated otherwise, and denote an edge between v and w as the concatenation of its endpoints, vw. Let $G=(V,E)$ denote a graph. A path P in G is denoted by listing its vertices in order, i.e., $P=v_1v_2\ldots v_\ell$. We use the following notation to refer to subpaths of a path P:

$$P_{[v_i,v_j]}=\begin{cases} v_iv_{i+1}\ldots v_j & \text{if } i\leq j,\\ v_iv_{i-1}\ldots v_j & \text{if } i>j.\end{cases}$$

By *disjoint* paths we will always mean *vertex-disjoint* paths. For a set of vertices $S\subseteq V$, we define $\delta(S)=\{vw\in E : v\in S, w\notin S\}$. For two sets X,Y their *symmetric difference* is denoted by $X\triangle Y=(X\setminus Y)\cup(Y\setminus X)$.

A set $C\subseteq V$ is a *vertex cover* of G if every edge of G has at least one endpoint in C; the *vertex cover number* $\tau(G)$ of G is the minimum cardinality of any vertex cover of G. An *independent set* of G is a set $S\subseteq V$ of pairwise nonadjacent vertices; the *independence number* $\alpha(G)$ of a graph G is the maximum cardinality of any independent set of G. The *maximum degree* and *minimum degree* of G are denoted by $\Delta(G)$ and $\delta(G)$ respectively. For a class $\mathcal{C}$ of graphs we define $\mathrm{d}_{\mathcal{C}}(G)=\min_{S\subseteq V: G-S\in\mathcal{C}}|S|$ to be the *vertex deletion distance* of G to $\mathcal{C}$; a set S such that $G-S\in\mathcal{C}$ is called a *modulator*. E.g., the vertex cover number $\tau(G)$ is the vertex deletion distance of G to edgeless graphs, i.e., independent sets.

A *matching* in a graph is a set of pairwise disjoint edges. Let $G=(V,E)$ be a graph and let $M\subseteq E$ be a matching in G. The matching M is *maximal* if there is no matching M' in G such that $M\subsetneq M'$ and M is *maximum* if there is no matching M' in G such that $|M|<|M'|$; the *matching number* $\nu(G)$ of G is the cardinality of a maximum matching in G.

A vertex $v\in V$ is called M-*matched* if there is an edge in M that contains v, and v is called M-*exposed* otherwise. We do not mention M if the matching is clear from the context. We say that an edge $e\in E$ is *blue* if $e\notin M$ and e is *red* if $e\in M$. An M-*alternating path* is a path in G that alternatingly uses red and blue edges. An M-*augmenting path* is an M-alternating path that starts and ends with an M-exposed vertex; a *shortest* M-augmenting path is an M-augmenting path that uses as few edges as possible.

It is well known that matchings can be enlarged along augmenting paths and that an augmenting path always exists if the matching is not maximum. We say that the matching $M \triangle E(P)$ is obtained by *augmenting* M along P.

▶ **Lemma 2.1.** *If M is a matching in G and P is an M-augmenting path, then $M \triangle E(P)$ is also a matching in G and has size $|M \triangle E(P)| = |M| + 1$.*

▶ **Theorem 2.2** ([29])**.** *Let M and N be matchings in $G = (V, E)$ with $|N| > |M|$. The subgraph $G' = (V, M \triangle N)$ of G contains at least $|N| - |M|$ vertex-disjoint M-augmenting paths.*

▶ **Corollary 2.3** ([7])**.** *A matching M is maximum if and only if there is no M-augmenting path.*

3 Hopcroft-Karp analysis

Many of the fastest algorithms for Maximum Matching make use of a framework introduced by Hopcroft and Karp [29] for the special case of bipartite matching. We give an overview of the framework in this section, mostly following Hopcroft and Karp [29]. The main idea is to search for shortest augmenting paths instead of arbitrary augmenting paths. Exhaustively searching for shortest augmenting paths and augmenting along them leads to Algorithm 1.

Algorithm 1: Generic Matching Algorithm.

```
  Input: Graph $G$
  Output: Maximum matching $M$
1 $M \leftarrow \emptyset$;
2 while $M$ is not maximum do
3     Find a shortest $M$-augmenting path $P$;
4     $M \leftarrow M \triangle E(P)$;
5 return $M$;
```

Let $P_1, P_2, \ldots, P_\ell$ be the sequence of augmenting paths in the order found during an execution of Algorithm 1. Hopcroft and Karp [29] observed the following properties of the computed shortest augmenting paths.

▶ **Lemma 3.1** ([29])**.** *Let M be a matching, let P be a shortest M-augmenting path, and let P' be a $M \triangle E(P)$-augmenting path, then $|P'| \geq |P| + 2|P \cap P'|$.*

▶ **Corollary 3.2** ([29])**.** *The sequence $|P_1|, \ldots, |P_\ell|$ is non-decreasing.*

▶ **Corollary 3.3** ([29])**.** *If $|P_i| = |P_j|$ for some $i \neq j$, then P_i and P_j are vertex-disjoint.*

Following these observations, we can partition the sequence $P_1, \ldots, P_\ell$ into maximal contiguous subsequences $P_i, P_{i+1}, \ldots, P_j$ such that $|P_i| = |P_{i+1}| = \cdots = |P_j| < |P_{j+1}|$, due to Corollary 3.3 the paths $P_i, P_{i+1}, \ldots, P_j$ must be pairwise vertex-disjoint. Every such subsequence is called a *phase* and corresponds to a maximal set of vertex-disjoint shortest augmenting paths due to Corollary 3.3. With the terminology of phases introduced, it is useful to restate Algorithm 1 as follows.

Algorithm 2: Phase Framework.

```
  Input: Graph $G$
  Output: Maximum matching $M$
1 $M \leftarrow \emptyset$;
2 while $M$ is not maximum do
3     Find a maximal set $S$ of vertex-disjoint shortest $M$-augmenting paths;
4     Augment $M$ along all paths in $S$;
5 return $M$;
```

In Algorithm 2 each iteration of the **while**-loop corresponds to a single phase. If an algorithm implements Algorithm 2 we say that it *employs the phase framework*. In the following, we will abstract from the implementation details of algorithms employing the phase framework and only bound the number of phases that are required in the worst case. Hopcroft and Karp [29] presented an upper bound in terms of the matching number $\nu(G)$.

▶ **Theorem 3.4** ([29]). *Every algorithm employing the phase framework requires at most* $2\left\lceil\sqrt{\nu(G)}\right\rceil + 2$ *phases.*

The next bound is a simple corollary of Theorem 3.4 by noticing that $\nu(G) \le \frac{n}{2}$, but we opt to give an independent proof to serve as an instructive example for the proofs to come.

▶ **Theorem 3.5** (folklore). *Every algorithm employing the phase framework requires at most* $\mathcal{O}(\sqrt{n})$ *phases.*

Proof. Let M denote the matching obtained after performing $\lceil\sqrt{n}\rceil$ phases of Algorithm 2. Every further M-augmenting path has length at least $\lceil\sqrt{n}\rceil$ by Corollary 3.2. This implies that we can pack at most $\lceil\sqrt{n}\rceil$ such augmenting paths into G and hence by Theorem 2.2, at most $\lceil\sqrt{n}\rceil$ augmentations remain. Since we perform at least one augmentation per phase, Algorithm 2 must have terminated after an additional $\lceil\sqrt{n}\rceil$ phases. Thus, Algorithm 2 terminates after at most $2\lceil\sqrt{n}\rceil$ phases. ◀

Several of the fastest Maximum Matching algorithms employ the phase framework [8, 27, 39]. Any one of these algorithms yields the following time bound for a single phase.

▶ **Theorem 3.6.** *There is an algorithm that given a matching M computes a maximal set of vertex-disjoint shortest M-augmenting paths in time $\mathcal{O}(m)$. In particular, each phase of the phase framework can be implemented to run in time $\mathcal{O}(m)$.*

With Theorem 3.5 and Theorem 3.4 we obtain the following time bounds.

▶ **Theorem 3.7.** *There is an algorithm employing the phase framework that solves* Maximum Matching *in time $\mathcal{O}(\sqrt{n}m)$ and $\mathcal{O}(\sqrt{\nu(G)}m)$.*

4 Adaptive parameterized analysis

In this section we will perform an adaptive analysis for algorithms employing the phase framework by analyzing the required number of phases in terms of various graph parameters. Theorem 3.6 yields an improved running time if the considered parameter is small enough.

Many of the considered parameters are NP-hard to compute, e.g., the vertex cover number. This is not an issue, however, as we only require the parameter value for the running time analysis and not for the execution of the algorithm. In this sense, algorithms employing the phase framework are proved to obliviously adapt to the studied parameters.

4.1 Short alternating paths

Our main lemma relies on bounding the length of shortest augmenting paths. In the interest of simplifying later arguments, we will not only bound the length of shortest augmenting paths, but also of alternating paths that are not necessarily augmenting. The strategy for obtaining such upper bounds is to take a long alternating path P and deduce that additional edges must exist in $G[V(P)]$ that enable us to find a shorter alternating path P' in $G[V(P)]$ between the endpoints of P. Hence, the replacement path P' will only visit vertices that are also visited by the original path P. The following definition formalizes this idea.

▶ **Definition 4.1.** Given a matching M in a graph G and an M-alternating path $P = v_1 \dots v_\ell$. We say that an M-alternating path $P' = w_1 \dots w_k$ *replaces* P if the following is true:

- $V(P') \subseteq V(P)$,
- $w_1 = v_1$ and $w_k = v_\ell$,
- P' has the same parity as P with respect to M, i.e.,
  $v_1v_2 \in M \iff w_1w_2 \in M$ and $v_{\ell-1}v_\ell \in M \iff w_{k-1}w_k \in M$.

In particular, if P' replaces P, then P' is at most as long as P.

A technicality that arises from considering general alternating paths, as opposed to augmenting paths, is that an alternating path that starts and ends with a blue edge, i.e. an edge not in the matching, might have endpoints that are not exposed. If we want to shortcut by taking a different edge incident to such an endpoint, then this edge might be red which causes our constructions to fail. To avoid this issue, it suffices to consider the subpath resulting from the removal of the first and last edge.

▶ **Definition 4.2.** A graph G is *ℓ-replaceable* if for every matching M each M-alternating path can be replaced by an M-alternating path of length at most ℓ. The class of ℓ-replaceable graphs is denoted $\mathcal{R}[\ell]$.

We will now show that algorithms employing the phase framework require only few phases for graphs that are close, in the sense of vertex deletion distance, to ℓ-replaceable graphs.

▶ **Lemma 4.3.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{k\ell})$ phases on graphs G with $\mathrm{d}_{\mathcal{R}[\ell]}(G) \leq k$. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{k\ell}m)$ for such graphs.*

Proof. Let $S \subseteq V$ with $|S| \leq k$, such that $G - S \in \mathcal{R}[\ell]$ and let M be the matching obtained after performing $\lceil\sqrt{k\ell}\rceil$ phases of Algorithm 2. We claim that every shortest M-augmenting path uses at least $\lfloor\sqrt{k}\rfloor$ vertices of S; every such path has a length of at least $\lceil\sqrt{k\ell}\rceil$ due to Corollary 3.2. Consider such a path P, since $G - S$ is ℓ-replaceable, we can assume that every time P enters $G - S$ it uses at most $\ell + 1$ vertices of $G - S$ before going back to S. Hence, P must use at least $\lfloor\sqrt{k}\rfloor - 1$ vertices of S to have a length of $\lceil\sqrt{k\ell}\rceil$ or more.

Since S is of size at most k and due to the properties of replacing paths, this implies that we can pack at most $2\lceil\sqrt{k}\rceil$ M-augmenting paths into G as

$$2\left\lceil\sqrt{k}\right\rceil\left(\left\lfloor\sqrt{k}\right\rfloor - 1\right) \geq 2\sqrt{k}\left(\sqrt{k} - 2\right) = 2k - 4\sqrt{k} \geq k \qquad \text{for } k \geq 16.$$

By Theorem 2.2 at most $2\lceil\sqrt{k}\rceil$ augmentations remain, which require at most $2\lceil\sqrt{k}\rceil$ phases. In total, we need $\lceil\sqrt{k\ell}\rceil + 2\lceil\sqrt{k}\rceil \in \mathcal{O}(\sqrt{k\ell})$ phases. Theorem 3.6 implies the time bound. ◀

The following running time bound relative to the vertex cover number $\tau(G)$ follows directly from Theorem 3.7 by the use of the well-known inequality $\nu(G) \leq \tau(G)$.

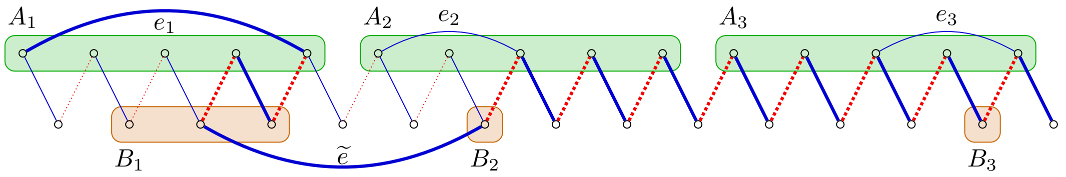


Figure 1 The construction in Theorem 4.6 for $\alpha(G) = 2$, the red dotted edges are matched and the blue edges are unmatched. The thickened edges represent the shorter alternating path P'.

▶ **Theorem 4.4.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{\tau(G)})$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{\tau(G)}m)$.*

Alternatively, observe that if $\mathcal{C}$ is the class of independent sets, then $\mathrm{d}_{\mathcal{C}}(G) = \tau(G)$ and hence Theorem 4.4 is implied by Lemma 4.3 as independent sets are trivially 1-replaceable.

More generally, every graph without paths of length $\ell + 1$ is ℓ-replaceable. It is known that a graph class has bounded path length if and only if it has bounded tree-depth (see, e.g., [40, Chapter 6]). As a further special case consider the class of *star forests*, i.e., graphs where every connected component is a star and therefore must be 2-replaceable. Hence, we obtain the following corollary of Lemma 4.3.

▶ **Corollary 4.5.** *Let $\mathcal{C}$ be the class of star forests. Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}}(G)})$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}}(G)}m)$.*

4.2 Independence number

A graph with independence number k contains many edges in the sense that any set of $k + 1$ vertices must induce an edge. We will use this property to shorten long alternating paths.

▶ **Theorem 4.6.** *Suppose that G is a graph such that $\alpha(G) \le k$, then G is $\mathcal{O}(k^2)$-replaceable.*

Proof. First, fix a matching M. We show how to replace alternating paths that begin and end with a blue edge, i.e., an edge not in M. By replacing appropriate subpaths of long alternating paths with other parities, the general result will follow.

Suppose that $P = a_1b_1a_2b_2 \dots a_\ell b_\ell$ is an alternating path that is longer than $4(k+1)^2+1$ and that starts and ends with a blue edge. We can assume that ℓ is odd. Distinguishing the vertices of P by their parity, we define $A = \{a_i : i \in [\ell]\}$ and $B = \{b_i : i \in [\ell]\}$. Furthermore, we define the sets A_i and A'_i for $i = 1, \dots, k+1$ by

$$A_i = \{a_{(i-1)(2k+2)+j} : j = 1, 2, 3, \dots, 2k+1\} \text{ and } A'_i = \{a_j \in A_i : j \text{ odd}\}.$$

Note that $|A'_i| = k + 1 > k$ for all i, hence the A'_i cannot be independent sets. Thus, there is at least one edge e_i in $G[A'_i]$. We denote the endpoints of e_i by

$$e_i = a_{p_i}a_{q_i}, \text{ where } p_i \le q_i - 2, \text{ for all } i = 1, \dots, k+1. \tag{1}$$

We now consider the vertices of B that lie between the endpoints of e_i on P; we omit b_{p_i} to ensure that the constructed path is shorter than P. Concretely, let $B_i = \{b_{p_i+1}, b_{p_i+2}, \dots, b_{q_i-1}\}$ for all $i = 1, \dots, k+1$. Equation (1) implies that $B_i \ne \emptyset$. Now, we arbitrarily choose a vertex b'_i from each B_i and define $B' = \{b'_i : i = 1, \dots, k+1\}$. Observe that B' cannot be an independent set as B' contains $k + 1$ vertices; thus, there must exist an edge $\tilde{e} = b'_ib'_j$ with $i < j$. We construct the path P' that replaces P by (see Figure 1)

$$P' = P_{[a_1,a_{p_i}]}P_{[a_{q_i},b'_i]}P_{[b'_j,b_\ell]},$$

using edges $a_{p_i}a_{q_i}$ and $b'_ib'_j$; note that $P_{[a_{q_i},b'_i]}$ is a subpath of P in reverse order. Note that both edges are blue because a_{q_i} and b'_i are already incident with red edges on P. It can be easily checked that P' is an alternating path and, in particular, that it is a valid replacement for P. We will now show that P' is strictly shorter than P. It suffices to compare the length of $P_1 = P_{[a_{p_i},b'_j]}$ and $P_2 = P'_{[a_{p_i},b'_j]} = a_{p_i}P_{[a_{q_i},b'_i]}b'_j$ as P and P' agree on the remaining parts. Let s and t be the indices such that $b'_i = b_s$ and $b'_j = b_t$. The length of P_1 is $2(t-p_i)+1$. For the length of P_2 we obtain

$$\begin{aligned}|P_2| &= 1 + (2(s-q_i)+1) + 1 = 2(s-q_i) + 3 < 2(s-(p_i+1)) + 3 = 2(s-p_i) + 1\\ &< 2(t-p_i) + 1 = |P_1|,\end{aligned}$$

where the first inequality follows from Equation (1). ◀

By combining this result with Lemma 4.3 we obtain the following corollary.

▶ **Corollary 4.7.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\alpha(G)^2)$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\alpha(G)^2 m)$.*

Let $\mathcal{C}_k$ denote the class of graphs with independence number at most k in each connected component. Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}_k}(G)}k^2)$ phases. In particular, Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{C}_k}(G)}k^2 m)$.*

A better analysis in terms of the independence number can be achieved by not using replaceability, but in exchange we lose the square root dependence on the size of the modulator.

▶ **Lemma 4.8.** *Every maximal matching M covers at least $n - \alpha(G)$ vertices.*

Proof. If $\alpha(G)+1$ vertices were exposed, they would not be an independent set and hence have an edge between them. Thus, M would not be maximal. ◀

▶ **Corollary 4.9.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(\alpha(G))$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\alpha(G)m)$.*

Proof. After the first phase, we know that at most $\alpha(G)$ vertices are exposed by Lemma 4.8. Hence, $\alpha(G)/2$ further augmentations suffice. ◀

4.3 s-plexes

An *s-plex*, $s \geq 1$, is an n-vertex graph G with minimum degree $\delta(G) \geq n - s$. They were introduced by Seidman and Foster [42] as a generalization of cliques, which are 1-plexes. Problems related to s-plexes have been studied in parameterized complexity before [28, 44]. Let $\mathcal{P}[s]$ denote the class of graphs that are disjoint unions of s-plexes. Given a vertex v and a set $W \subseteq V \setminus \{v\}$ of size at least s in an s-plex, we know that there is an edge vw for some $w \in W$. Thus, s-plexes allow for better control than a small independence number as we can guarantee the existence of an edge incident to some specific vertex instead of just getting some edge in a large set of vertices.

▶ **Theorem 4.10.** *If G is a k-plex, i.e., if $\delta(G) \geq n - k$, then G is $\mathcal{O}(k)$-replaceable.*

Proof. Let M be a matching in G and suppose that $P = v_1 \ldots v_\ell$ is an M-alternating path in G of length $\ell - 1 \geq 2k + 5$. Let i be the smallest integer such that v_i is M-exposed or $v_{i-1}v_i$ is red, i.e., $v_{i-1}v_i \in M$. Let j be the largest integer such that v_j is M-exposed or v_jv_{j+1} is red. We have $i \in \{1, 2, 3\}$ and $j \in \{\ell-2, \ell-1, \ell\}$.

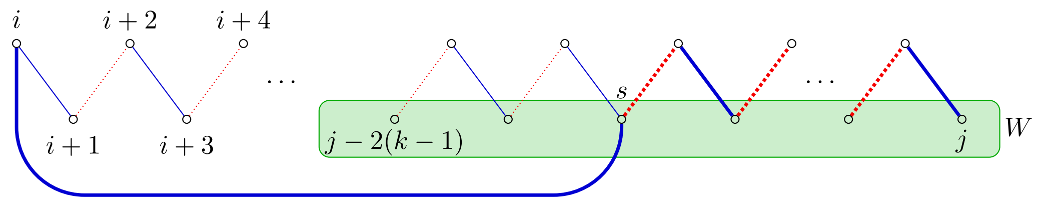


Figure 2 Replacing alternating paths in a k-plex.

Consider the vertices $W = \{v_j, v_{j-2}, v_{j-4}, \dots, v_{j-2(k-1)}\}$ and observe that $v_i \notin W$ as $3 < \ell - 2k$. The set W contains k vertices and since G is a k-plex there must be some $v_s \in W$ such that $v_i v_s \in E$. By choice of i and s the edges $v_i v_{i+1}$ and $v_{s-1} v_s$ must be blue. Similarly, by choice of i, the edge $v_i v_s$ is blue. Hence, as seen in Figure 2, we can replace P by the shorter M-alternating path $P' = P_{[v_1, v_i]} P_{[v_s, v_\ell]}$ with length $|P'| \leq 2 + 2(k-1) + 2 = 2k + 2$. ◀

By Lemma 4.3, we obtain the following corollary.

▶ **Corollary 4.11.** *Every algorithm employing the phase framework requires at most $\mathcal{O}(n - \delta(G))$ phases. In particular,* Maximum Matching *can be solved in time $\mathcal{O}((n - \delta(G))m)$.*

Every algorithm employing the phase framework requires at most $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{P}[s]}(G)s})$ phases. Hence, Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{\mathrm{d}_{\mathcal{P}[s]}(G)}sm)$. In particular,* Maximum Matching *can be solved in time $\mathcal{O}(\sqrt{k}m)$ on graphs that have distance at most k to cluster graphs.*

5 Lower bounds on the number of phases

In this section, we show that several restrictive graph classes do not admit results such as those obtained in the previous section. To this end, we show that the assumptions made about algorithms that follow the phase framework still allow a worst case of $\Omega(\sqrt{n})$ phases. In other words, further assumptions about the behavior of such an algorithm are necessary to avoid these lower bounds and to again get adaptive running times.

5.1 Paths and forests

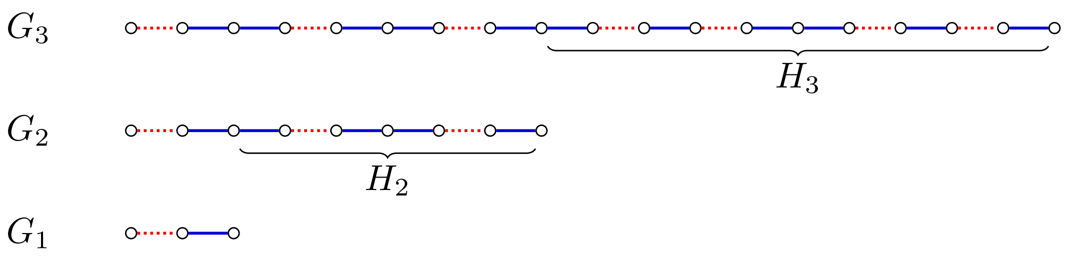


Figure 3 Lower bound construction for paths.

▶ **Lemma 5.1.** *An algorithm employing the phase framework may choose a sequence of augmentations resulting in at least $\Omega(\sqrt{n})$ phases on paths.*

Proof. We will concatenate increasingly long paths where we take every second edge into the matching in such a way that the matching on the newly added path is maximal but not maximum, so that every one of the concatenated paths requires a separate phase. More formally, on $P_{2i} = v_1 v_2 \dots v_{2i}$ we use the matching $\{v_{2k}v_{2k+1} : k \in [i-1]\}$. This matching is one edge away from the maximum and the only augmenting path has length $2i - 1$.

Let H_i be obtained by concatenating two copies of P_{2i} through identification of two endpoints (obtaining a path on $4i - 1$ vertices). The matching on H_i can be augmented once by an augmenting path of length $2i - 1$. There are two augmenting paths on H_i, we always choose to augment along the copy of P_{2i} that is attached to the previously constructed path, i.e., the left copy in Figure 3. Using two copies of P_{2i} in H_i ensures that every H_i requires at least one augmentation. This is not the case if we simply concatenate $P_2, P_4, \dots, P_{2k}$.

We can now define our desired paths. Let $G_1 = H_1 = P_3$ and in this exceptional case we assume that the left edge of P_3 is matched. Furthermore, let G_{i+1} be obtained from G_i by concatenating H_{i+1} at the right. The number of vertices of $G_{\lceil\sqrt{n}\rceil}$ can be bounded by

$$\sum_{i=1}^{\lceil\sqrt{n}\rceil} 2|V(P_{2i})| \leq 4(\lceil\sqrt{n}\rceil)^2 \in \mathcal{O}(n).$$

Now, we argue that our choice of augmentations leads to $\Omega(\sqrt{n})$ phases for $G_{\lceil\sqrt{n}\rceil}$. In the first phase we choose to find the maximal matching that we associated with each H_i. Now, observe that $G_{\lceil\sqrt{n}\rceil}$ contains augmenting paths of lengths $3, 5, \dots, 2\lceil\sqrt{n}\rceil - 1$. In phase i, $i \geq 2$ we augment along the left copy of P_{2i} in H_i. As this does not affect the augmenting paths in the other H_i, we need $\lceil\sqrt{n}\rceil$ phases in total. ◀

Using the parameter values on paths, we obtain the following parameterized lower bounds.

▶ **Corollary 5.2.** *An algorithm employing the phase framework may choose a sequence of augmentations resulting in* $\Omega(\sqrt{\alpha(G)})$, $\Omega(\sqrt{\tau(G)})$, $\Omega(\sqrt{\mathrm{d}_{\mathcal{P}[1]}(G)})$ *or* $\Omega(\sqrt{\mathrm{nd}(G)})$ *phases, where* $\mathrm{nd}(G)$ *is the neighborhood diversity of* G.

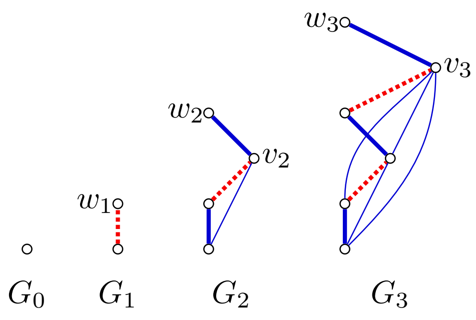
5.2 Cographs

Mertzios et al. [36] devised a MAXIMUM MATCHING algorithm parameterized by vertex deletion distance to cocomparability graphs and there are several MAXIMUM MATCHING algorithms parameterized by modular-width [11, 33]. In the interest of showing that these results cannot be replicated or improved by our approach, we give a lower bound result for cographs, i.e., the graphs of modular-width 2 and a subclass of cocomparability graphs.

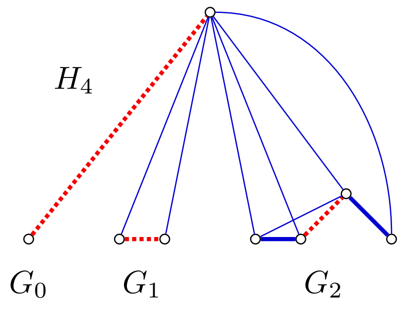
▶ **Definition 5.3.** A graph is a *cograph* if it can be constructed from the following operations:

- K_1 is a cograph,
- the disjoint union $G \cup H$ of two cographs G and H is a cograph,
- the join $G \times H$ of two cographs G and H is a cograph, where $V(G \times H) = V(G) \cup V(H)$ and $E(G \times H) = E(G) \cup E(H) \cup \{vw : v \in V(G), w \in V(H)\}$.

▶ **Theorem 5.4.** *There is a family of cographs such that an algorithm employing the phase framework may choose a sequence of augmentations resulting in* $\Omega(\sqrt{n})$ *phases on this family.*



(a) Increasingly long augmenting paths.



(b) Construction of H_4.

Figure 4 Cographs with increasingly long augmenting paths, the red dotted edges are matched and the blue edges are unmatched. The thickened edges denote the chosen augmenting paths.

Proof. We will construct appropriate cographs using the cograph operations. Let $G_0 = K_1$ and $G_1 = K_1 \times K_1 = P_2$ and for $i \geq 1$ define $G_{i+1} = (G_i \cup K_1) \times K_1$ as auxiliary graphs. Given n, we construct

$$H_n = \left(\bigcup_{i=0}^{\lceil \sqrt{n} \rceil} G_i \right) \times G_0.$$

Observe that G_i has at most $2i + 1$ vertices, therefore H_n has $\mathcal{O}(n)$ vertices. We will now describe a sequence of augmentations in H_n that requires $\Omega(\sqrt{n})$ phases.

In each G_i, $i \geq 2$, there is exactly one vertex of degree one, which we call w_i. For G_1, we fix an arbitrary vertex that is called w_1. Furthermore, the vertex that is joined last in the construction of G_i, $i \geq 2$, is referred to as v_i, see Figure 4. First, we describe which maximal matchings to associate with the graphs G_i. In G_0 there is no edge to choose, in G_1 we choose the only possible edge and in G_2 we take the edge $v_2 w_1$. Inductively, for G_{i+1}, $i \geq 2$, we use the maximal matching of G_i and add the edge $v_{i+1} w_i$. With H_n we associate the union of these matchings and add the edge between the two copies of G_0 to the matching. In this way we obtain the matching that is supposed to be found in the first phase.

We now argue that G_i, $i \geq 2$, has exactly one shortest augmenting path of length $2i - 1$. This is true for $i = 2$; the other cases follow by induction. Let P_i be the shortest augmenting path in G_i starting at w_i, then $P_{i+1} = w_{i+1} v_{i+1} P_i$ is an augmenting path of length $2i + 1$ in G_{i+1}. There are no further augmenting paths as there are only two exposed vertices in G_{i+1} and one of them is w_{i+1} with degree one; starting at w_{i+1}, we must first take the edge $w_{i+1} v_{i+1}$ and then the matched edge $v_{i+1} w_i$, hence we must take the path P_{i+1} by induction.

Let v denote the vertex in G_0 that is joined last in the construction of H_n. The construction of H_n does not create any additional augmenting paths as every maximal alternating path that passes through v must have one matched endpoint, namely the vertex in the other copy of G_0. In phase i we augment the shortest augmenting path of length $2i - 1$ in the copy of G_i. By repeating the previous argument, these augmentations cannot introduce any new augmenting paths in the later phases. Hence, we require $\lceil \sqrt{n} \rceil$ phases for this sequence of augmentations, thereby proving the claimed lower bound. ◀

The graphs in the previous proof are also C_4-free, therefore these graphs are not only cographs but also *trivially perfect graphs.*

6 Conclusion

We have conducted an adaptive analysis that applies to all algorithms for Maximum Matching that follow the phase framework of Hopcroft and Karp [29], such as the algorithms due to Micali and Vazirani [39], Blum [8], and Goldberg and Karzanov [27]. The main take-away message of our paper is that these algorithms not only obtain the best known time $\mathcal{O}(\sqrt{n}m)$ for solving Maximum Matching but that they are also (obliviously) adaptive to beneficial structure. That is, they run in linear time on several graph classes and they run in time $\mathcal{O}(\sqrt{k}m)$ for graphs that are k vertex deletions away from any of several classes; before, most bounds were $\Omega(km)$. Arguably, such adaptive algorithms are the best possible result for dealing with unknown beneficial structure because they are never worse than the general bound, in this case taking $\Omega(\sqrt{n}m)$ when $k = \Theta(n)$, and smoothly interpolate to linear time on well-structured instances. Moreover, in the present case, they unify several special cases and remove the need to find exact or approximate beneficial structure.

We complemented our findings by proving that the phase framework alone still allows taking $\Omega(\sqrt{n})$ phases, and, hence, total time $\Omega(\sqrt{n}m)$, even on restrictive classes like paths, trivially perfect graphs, and bipartite chain graphs (and their superclasses), despite the existence of (dedicated) linear-time algorithms. Of course, all of these cases are easy to handle but it raises the question whether there are simple further properties to demand of a phase-based algorithm so that it is provably adaptive to larger classes such as cocomparability or bounded treewidth graphs? In the same vein, it would be interesting whether time $\mathcal{O}(\sqrt{k}m)$ is possible relative to feedback vertex number k, i.e., relative to deletion distance to a forest.

More generally, with the large interest in "FPT in P" (or efficient parameterized algorithms), it seems interesting what other fundamental problems admit adaptive algorithms that interpolate between, say, linear time and the best general bound. Are there other cases where a proven algorithmic paradigm, like the path packing phases of Hopcroft and Karp [29], also obliviously yields the best known running times relative to beneficial input structure?

References

1 Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete Algorithms*, pages 377–391. SIAM, 2016.

2 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching Triangles and Basing Hardness on an Extremely Popular Conjecture. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 41–50. ACM, 2015. doi:10.1145/2746539.2746594.

3 Jérémy Barbay, Johannes Fischer, and Gonzalo Navarro. LRM-Trees: Compressed indices, adaptive sorting, and compressed permutations. *Theor. Comput. Sci.*, 459:26–41, 2012. doi:10.1016/j.tcs.2012.08.010.

4 Jérémy Barbay and Gonzalo Navarro. On compressing permutations and adaptive sorting. *Theor. Comput. Sci.*, 513:109–123, 2013. doi:10.1016/j.tcs.2013.10.019.

5 Holger Bast, Kurt Mehlhorn, Guido Schafer, and Hisao Tamaki. Matching algorithms are fast in sparse random graphs. *Theory of Computing Systems*, 39(1):3–14, 2006.

6 Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. In *International Symposium on Fundamentals of Computation Theory*, pages 96–110. Springer, 2017.

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

8 Norbert Blum. A new approach to maximum matching in general graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 586–597. Springer, 1990.

9 Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.76.

10 Maw-Shang Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In *International Symposium on Algorithms and Computation*, pages 146–155. Springer, 1996.

11 David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2765–2784. Society for Industrial and Applied Mathematics, 2018.

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

13 Yann Disser and Stefan Kratsch. Robust and Adaptive Search. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.26.

14 Feodor F. Dragan. On greedy matching ordering and greedy matchable graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 184–198. Springer, 1997.

15 Guillaume Ducoffe and Alexandru Popa. A quasi linear-time b-Matching algorithm on distance-hereditary graphs and bounded split-width graphs. *arXiv preprint*, 2018. arXiv:1804.09393.

16 Guillaume Ducoffe and Alexandru Popa. The use of a pruned modular decomposition for Maximum Matching algorithms on some graph classes. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, 29th International Symposium on Algorithms and Computation (ISAAC 2018), Jiaoxi, Yilan County, Taiwan, December 2018. doi:10.4230/LIPIcs.ISAAC.2018.144.

17 Jack Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.

18 Vladimir Estivill-Castro and Derick Wood. A Survey of Adaptive Sorting Algorithms. *ACM Comput. Surv.*, 24(4):441–476, 1992. doi:10.1145/146370.146381.

19 Till Fluschnik, Christian Komusiewicz, George B Mertzios, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. When can graph hyperbolicity be computed in linear time? In *Workshop on Algorithms and Data Structures*, pages 397–408. Springer, 2017.

20 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms (TALG)*, 14(3):34, 2018.

21 Jean-Luc Fouquet, Vassilis Giakoumakis, and Jean-Marie Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *International Journal of Foundations of Computer Science*, 10(04):513–533, 1999.

22 Jean-Luc Fouquet, Igor Parfenoff, and Henri Thuillier. An $\mathcal{O}(n)$ time algorithm for maximum matching in P_4-tidy graphs. *Information Processing Letters*, 62(6):281–287, 1997.

23 Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM (JACM)*, 38(4):815–853, 1991.

24 Frédéric Gardi. Efficient algorithms for disjoint matchings among intervals and related problems. In *Discrete Mathematics and Theoretical Computer Science*, pages 168–180. Springer, 2003.

25 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.

26 Fred Glover. Maximum matching in a convex bipartite graph. *Naval Research Logistics Quarterly*, 14(3):313–316, 1967.

27 Andrew V. Goldberg and Alexander V. Karzanov. Maximum skew-symmetric flows and matchings. *Mathematical Programming*, 100(3):537–568, 2004.

28 Jiong Guo, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. A more relaxed model for graph-based data clustering: s-plex editing. In *International Conference on Algorithmic Applications in Management*, pages 226–239. Springer, 2009.

29 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

30 Thore Husfeldt. Computing Graph Distances Parameterized by Treewidth and Diameter. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:11, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.IPEC.2016.16`.

31 Yoichi Iwata, Tomoaki Ogasawara, and Naoto Ohsaka. On the Power of Tree-Depth for Fully Polynomial FPT Algorithms. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 96. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

32 Leon Kellerhals. Parameterized Algorithms for Network Flows. Master's thesis, TU Berlin, 2018. URL: `https://fpt.akt.tu-berlin.de/publications/thesis/MA-leon-kellerhals.pdf`.

33 Stefan Kratsch and Florian Nelles. Efficient and Adaptive Parameterized Algorithms on Modular Decompositions. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ESA.2018.55`.

34 Y. Daniel Liang and Chongkye Rhee. Finding a maximum matching in a circular-arc graph. *Information Processing Letters*, 45(4):185–190, 1993.

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

36 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. Fine-grained algorithm design for matching. Technical report, Technical Report, 2016.

37 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. The Power of Linear-Time Data Reduction for Maximum Matching. In Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*, volume 83 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.MFCS.2017.46`.

38 George B. Mertzios, André Nichterlein, and Rolf Niedermeier. A Linear-Time Algorithm for Maximum-Cardinality Matching on Cocomparability Graphs. *SIAM Journal on Discrete Mathematics*, 32(4):2820–2835, 2018.

39 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, 1980*, pages 17–27. IEEE, 1980.

40 J. Nešetřil and P. Ossona de Mendez. *Sparsity (Graphs, Structures, and Algorithms), volume 28 of Algorithms and Combinatorics*. Springer, 2012.

41 Mihai Patrascu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010. `doi:10.1137/1.9781611973075.86`.

42 Stephen B. Seidman and Brian L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology*, 6(1):139–154, 1978.

43 George Steiner and Julian S Yeomans. A linear time algorithm for maximum matchings in convex, bipartite graphs. *Computers & Mathematics with Applications*, 31(12):91–96, 1996.

44 René Van Bevern, Hannes Moser, and Rolf Niedermeier. Approximation and tidying—a problem kernel for s-plex cluster vertex deletion. *Algorithmica*, 62(3-4):930–950, 2012.

45 Ming-Shing Yu and Cheng-Hsing Yang. An $\mathcal{O}$(n) time algorithm for maximum matching on cographs. *Information Processing Letters*, 47(2):89–93, 1993.

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

47 Raphael Yuster and Uri 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.