A Parameterized Study of Maximum Generalized Pattern Matching Problems

Sebastian Ordyniak¹ · Alexandru Popa²

Received: 14 October 2014 / Accepted: 30 April 2015 / Published online: 17 June 2015
© Springer Science+Business Media New York 2015

Abstract The generalized function matching (GFM) problem has been intensively studied starting with Ehrenfreucht and Rozenberg (Inf Process Lett 9(2):86–88, 1979). Given a pattern p and a text t, the goal is to find a mapping from the letters of p to non-empty substrings of t, such that applying the mapping to p results in t. Very recently, the problem has been investigated within the framework of parameterized complexity (Fernau et al. in FSTTCS, 2013). In this paper we study the parameterized complexity of the optimization variant of GFM (called Max-GFM), which has been introduced in Amir and Amihood (J Discrete Algorithms 5(3):514–523, 2007). Here, one is allowed to replace some of the pattern letters with some special symbols “?”, termed wildcards or don’t cares, which can be mapped to an arbitrary substring of the text. The goal is to minimize the number of wildcards used. We give a complete classification of the parameterized complexity of Max-GFM and its variants under a wide range of parameterizations, such as, the number of occurrences of a letter in the text, the size of the text alphabet, the number of occurrences of a letter in the pattern, the size of the pattern alphabet, the maximum length of a string matched to any pattern letter, the number of wildcards and the maximum size of a string that a wildcard can be mapped to.

Keywords (Maximum) generalized pattern matching · String morphisms · Parameterized complexity

¹ Institute for Computer Graphics and Algorithms, TU Wien, Vienna, Austria
² School of Science and Technology, Nazarbayev University, Astana, Kazakhstan
1 Introduction

In the generalized function matching problem one is given a text $t$ and a pattern $p$ and the goal is to decide whether there is a match between $p$ and $t$, where a single letter of the pattern is allowed to match multiple letters of the text (we say that $p$ GF-matches $t$). For example, if the text is $t = xyyx$ and the pattern is $p = aba$, then a generalized function match (on short, GF-match) is $a \rightarrow x$, $b \rightarrow yy$, but if $t = xyyz$ and $p = aba$, then there is no GF-match. If, moreover, the matching is required to be injective, then we term the problem generalized parameterized matching (GPM).

In [1], Amir and Nor describe applications of GFM in various areas such as software engineering, image searching, DNA analysis, poetry and music analysis, or author validation. GFM is a computational problem whose variants appear in areas such as avoidable or unavoidable patterns [12], word equations [13] and the ambiguity of morphisms [11].

GFM has a long history starting from 1979. Ehrenfeucht and Rozenberg [7] show that GFM is NP-complete. Independently, Angluin [2,3] showed, at the same time as Rozenberg and Salomaa, but independently, that GFM is NP-complete. Angluin’s paper received a lot of attention, especially in language theory and learning theory [16, 17, 20] (see [14] for a survey) but also in many other areas.

Recently, a systematic study of the classical complexity of a number of variants of GFM and GPM under various restrictions has been carried out [8,18,19]. It was shown that GFM and GPM remain NP-complete for many natural restrictions. Moreover, the study of GFM and its variants within the framework of parameterized complexity has recently been initiated [9].

In this paper we study the parameterized complexity of the optimization variant of GFM (called Max-GFM) and its variants, where one is allowed to replace some of the pattern letters with some special symbols “?”, termed wildcards or don’t cares, which can be mapped to an arbitrary substring of the text. The goal is to minimize the number of wildcards used. The problem was first introduced to the pattern matching community by Amir and Nor [1]. They show that if the pattern alphabet has constant size, then a polynomial algorithm can be found, but that the problem is NP-complete otherwise. Then, in [4], it is shown the NP-hardness of the GFM (without wildcards) and the NP-hardness of the GFM when the function $f$ is required to be an injection (named GPM). More specifically, GFM is NP-hard even if the text alphabet is binary and each letter of the pattern is allowed to map to at most two letters of the text [4]. In the same paper it is given a $\sqrt{OPT}$ approximation algorithm for the optimization variant of GFM where the goal is to search for a pattern $p'$ that GF-matches $t$ and has the smallest Hamming distance to $p$. In [5] the optimization versions of GFM and GPM are proved to be APX-hard.

Our results Before we discuss our results, we give formal definitions of the problems. In the following let $t$ be a text over an alphabet $\Sigma_t$ and let $p = p_1 \ldots p_m$ be a pattern over an alphabet $\Sigma_p$. We say that $p$ GF-matches $t$ if there is a function $f : \Sigma_p \rightarrow \Sigma_t$ such that $f(p_1) \ldots f(p_m) = t$. To improve the presentation we will sometimes abuse notation by writing $f(p)$ instead of $f(p_1) \ldots f(p_m)$. Let $k$ be a natural number. We say that a pattern $p$ $k$-GF-matches $t$ if there is a pattern $p'$ over alphabet $\Sigma_p \cup \{?,...,?_k\}$ such that $p'$ GF-matches $t$ and $p'$ can be obtained...
from $p$ after replacing at most $k$ occurrences of letters in $p$ with any of the wildcards $?, \ldots, ?^k$.

**Problem 1** (Maximum Generalized Function Matching) Given a text $t$, a pattern $p$, and an integer $k$, decide whether $p$ $k$-GF-matches $t$.

The Max-GFM can be seen as the optimization variant of GFM in which we want to replace some of the pattern letters with special wildcard symbols, i.e., the symbols $?, \ldots, ?^k$, which can be mapped to any non-empty substring of the text.

We also study the Max-GPM problem. The only difference between Max-GPM and Max-GFM is that for Max-GPM the function $f$ is required to be injective. The notions of GP-matching and $k$-GP-matching are defined in the natural way, e.g., we say a pattern $p$ GP-matches a text $t$ if $p$ GF-matches $t$ using an injective function.

In this paper we study the parameterized complexity of the two problems using a wide range of parameters:

1. maximum number of occurrences of a letter in the text $\#\Sigma_t$;
2. maximum number of occurrences of a letter in the pattern $\#\Sigma_p$;
3. size of the text alphabet $|\Sigma_t|$;
4. size of the pattern alphabet $|\Sigma_p|$;
5. the maximum length of a substring of the text that a letter of the pattern alphabet can be mapped to (i.e., $\max_i |f(p_i)|$);
6. the number of wildcard letters $\#?$;
7. the maximum length of a substring of the text that a wildcard can be mapped to, denoted by $\max |f(?)|$.

Our results are summarized in Table 1. We verified the completeness of our results using a simple computer program. In particular, the program checks for every of the 128 possible combinations of parameters $\mathcal{C}$ that the table contains either: (i) a superset of $\mathcal{C}$ under which Max-GFM/GPM is hard (and thus, Max-GFM/GPM is hard if parameterized by $\mathcal{C}$); or (ii) a subset of $\mathcal{C}$ for which Max-GFM/GPM is fpt (and then we have an fpt result for the set of parameters $\mathcal{C}$). Since some of our results do not hold for both Max-GFM and Max-GPM, we carried out two separate checks, one for Max-GFM and one for Max-GPM.

The paper is organized as follows. In Sect. 2 we give preliminaries, in Sect. 3 we present our fixed-parameter algorithms and in Sect. 4 we show our hardness results.

2 Preliminaries

We define the basic notions of Parameterized Complexity and refer to other sources [6, 10] for an in-depth treatment. A *parameterized problem* is a set of pairs $\langle I, k \rangle$, the *instances*, where $I$ is the main part and $k$ the *parameter*. The parameter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable (fpt)* if there exists an algorithm that solves any instance $\langle I, k \rangle$ of size $n$ in time $f(k)n^c$ where $f$ is an arbitrary computable function and $c$ is a constant independent of both $n$ and $k$. FPT is the class of all fixed-parameter tractable parameterized problems. Because we focus on fixed-parameter tractability of a problem we will sometimes use the notation
Table 1  Parameterized complexity of Max-GFM and Max-GPM

| #Σι | |Σι| | #Σρ | |Σρ| | max|f(pι)| | #? | max|f(?)| | Complexity |
|------|------|------|------|------|------|--------|--------|------|--------|------|------|
| par  | par  | –    | –    | –    | –    | –      | –      | –    | –      | par  | FPT (Corollary 3) |
| –    | par  | –    | par  | par  | –    | –      | FPT (Theorem 1) |
| –    | par  | –    | –    | par  | –    | FPT only GPM (Corollary 1) |
| –    | –    | par  | par  | par  | –    | FPT (Corollary 2) |
| –    | –    | –    | par  | par  | par  | FPT (Theorem 2) |
| par  | –    | par  | par  | par  | par  | par    | W[1]-h (Theorem 6) |
| par  | –    | par  | –    | par  | par  | par    | W[1]-h (Theorem 5) |
| par  | –    | par  | –    | –    | par  | par    | W[1]-h (Theorem 4) |
| –    | par  | par  | –    | par  | par  | –      | para-NP-h (Theorem 2) |
| –    | par  | –    | par  | –    | –    | –      | para-NP-h only GFM (8) |

O* to suppress exact polynomial dependencies, i.e., a problem with input size $n$ and parameter $k$ can be solved in time $O^*(f(k))$ if it can be solved in time $O(f(k)n^c)$ for some constant $c$.

Parameterized complexity offers a completeness theory, similar to the theory of NP-completeness, that allows the accumulation of strong theoretical evidence that some parameterized problems are not fixed-parameter tractable. This theory is based on a hierarchy of complexity classes $\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \text{W}[3] \subseteq \cdots$ where all inclusions are believed to be strict. An fpt-reduction from a parameterized problem $P$ to a parameterized problem $Q$ is a mapping $R$ from instances of $P$ to instances of $Q$ such that (i) $(I, k)$ is a Yes-instance of $P$ if and only if $(I', k') = R(I, k)$ is a Yes-instance of $Q$, (ii) there is a computable function $g$ such that $k' \leq g(k)$, and (iii) there is a computable function $f$ and a constant $c$ such that $R$ can be computed in time $O(f(k) \cdot n^c)$, where $n$ denotes the size of $(I, k)$.

For our hardness results we will often reduce from the following problem, which is well-known to be $\text{W}[1]$-complete [15].

**Multicolored Clique**

*Instance:* A $k$-partite graph $G = \langle V, E \rangle$ with a partition $V_1, \ldots, V_k$ of $V$.

*Parameter:* The integer $k$.

*Question:* Are there nodes $v_1, \ldots, v_k$ such that $v_i \in V_i$ and $\{v_i, v_j\} \in E$ for all $i$ and $j$ with $1 \leq i < j \leq k$ (i.e. the subgraph of $G$ induced by $\{v_1, \ldots, v_k\}$ is a clique of size $k$)?

For our hardness proofs we will often make the additional assumptions that (1) $|V_i| = |V_j|$ for every $i$ and $j$ with $1 \leq i < j \leq k$ and (2) $|E_{i,j}| = |E_{r,s}|$ for every $i$, $j$, $r$, and $s$ with $1 \leq i < j \leq k$ and $1 \leq r < s \leq k$, where $E_{i,j} = \{\{u, v\} \in E \mid u \in V_i$ and $v \in V_j\}$ for every $i$ and $j$. To see that **Multicolored Clique** remains...
W[1]-hard under these additional restrictions we can reduce from MULTICOLORED CLIQUE to its more restricted version using a simple padding construction as follows. Given an instance $\langle G, k \rangle$ of MULTICOLORED CLIQUE we construct an instance of its more restricted version by adding edges (whose endpoints are new vertices) between parts (i.e. $V_1, \ldots, V_k$) that do not already have the maximum number of edges between them and then adding isolated vertices to parts that do not already have the maximum number of vertices.

Even stronger evidence that a parameterized problem is not fixed-parameter tractable can be obtained by showing that the problem remains NP-complete even if the parameter is a constant. The class of these problems is called para-NP.

A square is a string consisting of two copies of the same (non-empty) string. We say that a string is square-free if it does not contain a square as a substring.

3 Fixed-Parameter Tractable Variants

In this section we show our fixed-parameter tractability results for Max-GFM and Max-GPM. In particular, we show that Max-GFM and Max-GPM are fixed-parameter tractable parameterized by $|\Sigma_t|$, $|\Sigma_p|$, and $\max_i |f(p_i)|$, and also parameterized by $\#?$, $\max_i |f(?)|$, $|\Sigma_p|$, and $\max_i |f(p_i)|$. We start by showing fixed-parameter tractability for the parameters $|\Sigma_t|$, $|\Sigma_p|$, and $\max_i |f(p_i)|$. We need the following lemma.

**Lemma 1** Given a pattern $p = p_1 \ldots p_m$ over an alphabet $\Sigma_p$, a text $t = t_1 \ldots t_n$ over an alphabet $\Sigma_t$, a natural number $q$, and a function $f : \Sigma_p \to \Sigma^+_t$, then there is a polynomial time algorithm deciding whether $pq$-GF/GP-matches $t$ using the function $f$.

**Proof** If we are asked whether $pq$-GP-matches $t$ and $f$ is not injective, then we obviously provide a negative answer. Otherwise, we use a dynamic programming algorithm that is similar in spirit to an algorithm in [4]. Let $\Sigma_p = \{a_1, \ldots, a_k\}$. For every $1 \leq i \leq m$ and $1 \leq j \leq n$, we define the function $g(i, j)$ to be the Hamming GFM/GPM-similarity (i.e., $m$ minus the minimum number of wildcards needed) between $t_1 t_2 \ldots t_j$ and $p_1 p_2 \ldots p_i$. Then, we obtain the Hamming GFM/GPM-similarity between $p$ and $t$ as $g(m, n)$. Consequently, if $m - g(m, n) > q$, we return No, otherwise we return Yes.

We now show how to recursively compute $g(i, j)$. If $i = 0$, we set $g(i, j) = 0$ and if $i \leq j$, we set:

$$g(i, j) = \max_{1 \leq k \leq j} \{g(i - 1, j - k) + I(t_{j-k+1} \ldots t_j, f(p_i))\}$$

where $I(s_1, s_2)$ is 1 if the strings $s_1$, and $s_2$ are the same, and 0 otherwise.

We must first show that the dynamic programming procedure computes the right function and then that it runs in polynomial time. We can see immediately that $g(0, i) = 0$ for all $i$ because in this case the pattern is empty. The recursion step of $g(i, j)$ has two cases: If $t_j - |f(p_i)| + 1 \ldots t_j = f(p_i)$, then it is possible to map $p_i$ to $f(p_i)$, and we can increase the number of mapped letters by one. Otherwise, we cannot increase
the Hamming GFM/GPM-similarity. However, we know that \( p_i \) has to be set to a wildcard and therefore we find the maximum of the previous results for different length substrings that the wildcard maps to.

It is straightforward to check that \( g(m, n) \) can be computed in cubic time.

**Theorem 1** Max-GFM and Max-GPM parameterized by \(|\Sigma_i|, |\Sigma_p|, \) and \( \max_i |f(p_i)| \) are fixed-parameter tractable.

**Proof** Let \( p, t, \) and \( q \) be an instance of Max-GFM or Max-GPM, respectively. The pattern \( p \) \( q \)-GF/GP-matches \( t \) if and only if there is a function \( f : \Sigma_p \rightarrow \Sigma_i^+ \) such that \( p \) \( q \)-GF/GP-matches \( t \) using \( f \). Hence, to solve Max-GFM/Max-GPM, it is sufficient to apply the algorithm from Lemma 1 to every function \( f : \Sigma_p \rightarrow \Sigma_i^+ \) that could possibly contribute to a \( q \)-GF/GP-matching from \( p \) to \( t \). Because there are at most \((|\Sigma_i|)^{\max_i |f(p_i)|} |\Sigma_p| \) such functions \( f \) and the algorithm from Lemma 1 runs in polynomial time, the running time of this algorithm is \( O^*((|\Sigma_i|)^{\max_i |f(p_i)|} |\Sigma_p|) \), and hence fixed-parameter tractable in \(|\Sigma_i|, |\Sigma_p|, \) and \( \max_i |f(p_i)| \). \( \square \)

Because in the case of Max-GPM it holds that if \( |\Sigma_i| \) and \( \max_i |f(p_i)| \) is bounded then also \( |\Sigma_p| \) is bounded by \(|\Sigma_i|^{\max_i |f(p_i)|} \), we obtain the following corollary.

**Corollary 1** Max-GPM parameterized by \(|\Sigma_i| \) and \( \max_i |f(p_i)| \) is fixed-parameter tractable.

We continue by showing our second tractability result for the parameters \(|\Sigma_p|, \max_i |f(p_i)|, \#?, \) and \( \max |f(?)| \).

**Theorem 2** Max-GFM and Max-GPM parameterized by \(|\Sigma_p|, \max_i |f(p_i)|, \#?, \) and \( \max |f(?)| \), are fixed-parameter tractable.

**Proof** Let \( p, t, \) and \( q \) be an instance of Max-GFM or Max-GPM, respectively.

Observe that if we could go over all possible functions \( f : \Sigma_p \rightarrow \Sigma_i^+ \) that could possibly contribute to a \( q \)-GF/GP-matching from \( p \) to \( t \), then we could again apply Lemma 1 as we did in the proof of Theorem 1. Unfortunately, because \(|\Sigma_i| \) is not a parameter, the number of these functions cannot be bounded as easily anymore. However, as we will show next it is still possible to bound the number of possible functions solely in terms of the parameters. In particular, we will show that the number of possible substrings of \( t \) that any letter of the pattern alphabet can be mapped to is bounded by a function of the parameters. Because also \(|\Sigma_p| \) is a parameter this immediately implies a bound (only in terms of the given parameters) on the total number of these functions.

Let \( c \in \Sigma_p \) and consider any \( q \)-GF/GP-matching from \( p \) to \( t \), i.e., a text \( p' = p'_1 \ldots p'_m \) of Hamming distance at most \( q \) to \( p \) and a function \( f : \Sigma_p \cup \{?, \ldots, ?_q \} \rightarrow \Sigma_i^+ \) such that \( f(p'_1) \ldots f(p'_m) = t \). Then either \( c \) does not occur in \( p' \) or \( c \) occurs in \( p' \). In the first case we can assign to \( c \) any non-empty substring over the alphabet \( \Sigma_i \) (in the case of Max-GPM one additionally has to ensure that the non-empty substrings over \( \Sigma_i \) that one chooses for distinct letters in \( \Sigma_p \) are distinct). In the second case let \( p'_i \) for some \( i \) with \( 1 \leq i \leq m \) be the first occurrence of \( c \) in \( p' \), let \( \overline{p}l_{i-1} = p'_1 \ldots p'_{i-1} \), and let \( \overline{p}l_{i-1} = p_1 \ldots p_{i-1} \). Furthermore, for every \( b \in \Sigma_p \cup \{?, \ldots, ?_q \} \) and \( w \in
and for cs parameterized reduction from show hardness for the different variants of (Max-)GFM and (Max-)GPM, we use a

4 Hardness Results

Algorithmica (2016) 75:1–26 7

Since |Σp| and #Σp together bound #?, we obtain the following corollary.

Corollary 2 Max-GFM and Max-GPM parameterized by #Σp, |Σp|, max_i |f(p_i)|, and max |f(?)| are fixed-parameter tractable.

Furthermore, because all considered parameters can be bounded in terms of the parameters #Σi and |Σi|, we obtain the following corollary as a consequence of any of our above fpt-results.

Corollary 3 Max-GFM and Max-GPM parameterized by #Σi and |Σi| are fixed-parameter tractable.

(|Σ_p ∪ {?1, ..., ?q})*, we denote by #b, w the number of times b occurs in w. Then

f(c) = t_{c_{i+1}} \ldots t_{c_{i+|f(c)|}} where c_s = \sum_{j=1}^{i-1} |f(p'_j)|, which implies that the value of f(c) is fully determined by c_s and |f(c)|. Because the number of possible values for |f(c)| is trivially bounded by the parameters (it is bounded by maxi |f(p_i)|), it remains to show that also c_s is bounded by the given parameters.

Because c_s = \sum_{j=1}^{i-1} |f(p'_j)| = (\sum_{b \in \Sigma_p \cup {?1, ..., ?q}} #b, \overline{p}_{i-1} |f(b)|), we obtain that the value of c_s is fully determined by the values of #b, \overline{p}_{i-1} and |f(b)| for every b ∈ Σ_p ∪ {?1, ..., ?q}. For every ? ∈ {?1, ..., ?q} there are at most 2 possible values for #(?), \overline{p}_{i-1} (namely 0 and 1) and there are at most max |f(?)| possible values for |f(?)|. Similarly, for every b ∈ Σ_p there are at most q + 1 possible values for #b, \overline{p}_{i-1} (the values #b, \overline{p}_{i-1} − q, ..., #b, \overline{p}_{i-1}) and there are at most max_i |f(p_i)| possible values for |f(b)|. Hence, the number of possible values for c_s is bounded in terms of the parameters, as required.

In this section we give our hardness results for (Max-)GFM and (Max-)GPM. To show hardness for the different variants of (Max-)GFM and (Max-)GPM, we use a parameterized reduction from MULTICOLORED CLIQUE. For the remainder of this section we will assume that we are given an instance of MULTICOLORED CLIQUE, i.e., a k-partite graph G = (V, E) with partition V_1, ..., V_k of V. For every i and j with 1 ≤ i < j ≤ k, we denote by E_{i,j} the set {u, v} ∈ E | u ∈ V_i and v ∈ V_j}. As stated in the preliminaries we can assume that |V_i| = n and |E_{i,j}| = m for every i and j with 1 ≤ i < j ≤ k. We will also assume that V_i = {v_1^i, ..., v_n^i} and E_{i,j} = {e_1^{i,j}, ..., e_m^{i,j}}. For a vertex v ∈ V and j with 1 ≤ j ≤ k we denote by E_j(v) the set of edges of G that are incident to v and whose other endpoint is in V_j.

We will employ two main types of reductions: (1) vertex-type reductions, where we “guess” the vertices of a k-clique of G and (2) edge-type reductions, where we “guess” the edges of a k-clique of G.

In the vertex-type reductions the pattern alphabet contains the letters V_1, ..., V_k and the reduction ensures that the set of vertices mapped to V_1, ..., V_k corresponds to a k-clique of G. This is achieved by forcing that for every pair i and j with 1 ≤ i < j ≤ k, it holds that the vertices corresponding to the text mapped to V_i and V_j are the endpoints of an edge in E_{i,j}.

© Springer
In the edge-type reductions the pattern alphabet contains the letters $E_{i,j}$ for every $i$ and $j$ with $1 \leq i \leq k$ and $1 \leq j \leq k$. The reduction then ensures that each $E_{i,j}$ is mapped to some text representing an edge in $E_{i,j}$ and that for every $i$ with $1 \leq i \leq k$, all edges corresponding to the text mapped to any $E_{i,j}$ with $j \neq i$ have the same endpoint in $V_i$. This then implies that the edges corresponding to the text mapped to all $E_{i,j}$ form a $k$-clique of $G$.

A common theme for all our reductions is that we often need to ensure that a certain substring of the pattern is mapped to exactly one of a list of substrings of the text. For instance, let's say we want that the letter $p$ substring of the pattern is mapped to exactly one of a list of substrings of the text. If we want to “place” $p$ substring of length two to the left and right of $p$ substring of length one or two. By choosing the number of letters that are mapped to a (sometimes distinct) letters such that each of these letters can be either mapped to a $E$ or $R$ from $E$ all edges corresponding to the text mapped to any $E_i$ has the same separator $t$, respectively. Varying the lengths of the text to which the “dummy” letters $L$ and $R$ are mapped to now allows one to map $p$ to any of the letters $t_1, \ldots, t_n$. Note that this way it would also be possible to map $p$ to any substring of $; t_1; t_2; \ldots; t_n; ;$ in the text as well as a substring of the form $#L; p; R#$ in the pattern. Here, # and ; are special separator letters (in both the text and the pattern) such that the pattern letters # and ; always have to be mapped to the text letters # and ;, respectively. Varying the lengths of the text to which the “dummy” letters $L$ and $R$ are mapped to now allows one to map $p$ to any of the letters $t_1, \ldots, t_n$. Note that this way it would also be possible to map $p$ to any substring of $; t_1; t_2; \ldots; t_n; ;$, however, any substring containing more than exactly one of $t_1, \ldots, t_n$ also necessarily contains the separator $;$ and the complete reduction ensures that the remaining occurrences of $p$ cannot be mapped any more. Note that the details of the above construction vary depending on the considered parameters in the reduction. For instance, if either $\max_i |f(p_i)|$ or $\max |f(?)|$ are bounded, the construction has to be adapted since it uses the property that $L$ and $R$ (or their wildcard replacements) can be mapped to arbitrary long substrings of the text. In these cases $L$ and $R$ are replaced by many (sometimes distinct) letters such that each of these letters can be either mapped to a substring of length one or two. By choosing the number of letters that are mapped to a substring of length two to the left and right of $p$ appropriately, it is then still possible to “place” $p$ over the required substring of the text.

To simplify and shorten the constructions, we will often use the following shortcuts:

- For an edge $e \in E$ between $v^l_i$ and $v^l_j$ where $1 \leq i < j \leq k$ and $1 \leq l, s \leq n$, we write $vt(e)$ to denote the text $v^l_i \oplus v^l_j$.
- For a letter $l$ of the text or pattern alphabet and $i \in \mathbb{N}$, we write $rp(l, i)$ to denote the text consisting of repeating the letter $l$ exactly $i$ times.
- For a symbol $l$, we write $enu(l, i)$ to denote the text $l_1 \cdots l_i$.

To ease understanding, we decided to present our hardness results in increasing level of difficulty. The first two theorems use a reduction of the vertex-type with the first vertex-type reduction being slightly simpler than the second. The remaining theorems use a reduction of the edge-type and are again presented in increasing levels of difficulty.

**Theorem 3** Max-GFM and Max-GPM are $W[1]$-hard parameterized by $\# \Sigma_p$, $|\Sigma_p|$, $\max_i |f(p_i)|$, and $\#$? (even if $\max_i |f(p_i)| \leq 1$).

Let $k' = 2^k$. We will show the theorem by constructing a text $t$ and a pattern $p$ from $G$ and $k$ in polynomial time such that:

(C1) the parameters $\# \Sigma_p$, $|\Sigma_p|$, and $\#$ can be bounded by a function of $k$. 

\textcopyright Springer
(C2) The following statements are equivalent:
1. $p$ is a $k'$-GF/GP-matches $t$;
2. $p$ is a $k'$-GF/GP-matches $t$ using a function $f$ with $\max_{p \in \Sigma_p} |f(p)| \leq 1$;
3. $G$ has a $k$-clique.

We set \( \Sigma_t = \{ ;, @, #, \% \} \cup \{ l_{i,j}, r_{i,j} \mid 1 \leq i < j \leq k \} \cup \{ v_i^j \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq k \} \) and \( \Sigma_p = \{ ;, @, #, D \} \cup \{ V_i \mid 1 \leq i \leq k \} \).

We define a preliminary text \( t' \) as follows.

\[
\#l_{1,2}; vt(e_{1,2}^1); \cdots; vt(e_{1,2}^m); r_{1,2}; \cdots; \#l_{1,k}; vt(e_{1,k}^1); \cdots; vt(e_{1,k}^m); r_{1,k} \\
\#l_{2,3}; vt(e_{2,3}^1); \cdots; vt(e_{2,3}^m); r_{2,3}; \cdots; \#l_{2,k}; vt(e_{2,k}^1); \cdots; vt(e_{2,k}^m); r_{2,k} \\
\cdots \\
\#l_{k-1,k}; vt(e_{k-1,k}^1); \cdots; vt(e_{k-1,k}^m); r_{k-1,k} \\
\]

We also define a preliminary pattern \( p' \) as follows.

\[
\#; V_1 \oplus V_2; D \# \cdots D; V_1 \oplus V_k; D \\
\#; V_2 \oplus V_3; D \# \cdots D; V_2 \oplus V_k; D \\
\cdots \\
\#; V_{k-1} \oplus V_k; D 
\]

Informally, the text \( t' \) and the pattern \( p' \) ensure that for every pair \( i \) and \( j \) with \( 1 \leq i < j \leq k \), it holds that the vertices corresponding to the letters mapped to \( V_i \) and \( V_j \) are the endpoints of an edge in \( E_{ij} \), which implies that the set of vertices corresponding to the letters mapped to \( V_1, \ldots, V_k \) is a $k$-clique of \( G \). To achieve this we also need the text \( t'' \) and the pattern \( p'' \) (defined below) which ensure that the separator letters ;, @, and \# of the pattern must be mapped to the corresponding separator letters of the text and also that the letter \( D \) in \( \Sigma_p \) is mapped to \( \% \) in \( \Sigma_t \) which in turn forces that the occurrences of \( D \) in \( p' \) are replaced by wildcards.

Let \( q = 2(k' + 1) \). Then \( t' \) is obtained by preceding \( t' \) with the text \( t'' \) defined as follows.

\[
\#; @\text{rp}(\%, q) 
\]

Similarly, \( p \) is obtained by preceding \( p' \) with the text \( p'' \) defined as follows.

\[
\#; @\text{rp}(D, q) 
\]

This completes the construction of \( t \) and \( p \). Clearly, \( t \) and \( p \) can be constructed from \( G \) and \( k \) in fpt-time (even polynomial time). Furthermore, because \( \# \Sigma_p = q + k' = 2(k' + 1) + k' = 3k' + 1, |\Sigma_p| = k + 4, \text{ and } \#? = k', \) condition (C1) above is satisfied. To show the remaining condition (C2), we need the following intermediate lemmas.

**Lemma 2** If \( G \) has a \( k \)-clique, then \( p \) is a \( k' \)-GF/GP-matches to \( t \) using a function \( f \) with \( \max_{p \in \Sigma_p} |f(p)| \leq 1 \).
Proof Because every GP-matching is also a GF-matching, it is sufficient to show that $p$ $k'$-GP-matches to $t$.

Let $\{v_{h_{1}}, \ldots, v_{h_{k}}\}$ be the vertices and $\{e_{i,j}^{i,j} \mid 1 \leq i < j \leq k\}$ be the edges of a $k$-clique of $G$ with $1 \leq h_{j} \leq n$ and $1 \leq h_{i,j} \leq m$ for every $i$ and $j$ with $1 \leq i < j \leq k$. We put $k'$ wildcards on the last $k'$ occurrences of $D$ in $p$. Informally, these wildcards are mapped in such a way that for every $1 \leq i < j \leq k$ the substring $\{V_{i} \oplus V_{j}\}$ of the pattern $p$ is mapped to the substring $\{v_{h_{i}}; v_{h_{j}}\}$ of the text $t$. More formally, for $i$ and $j$ with $1 \leq i < j \leq k$ let $q = (\sum_{a=1}^{\alpha} (k - a)) + j$. We map the wildcard on the $2(q - 1)$th occurrence of the letter $D$ in $p'$ with the text $\{v_{1}; v_{2}; \ldots; v_{m}\}$ and similarly we map widecards on the $(2(q - 1) + 1)$th occurrence of the letter $D$ in $p'$ with the text $\{v_{1}; v_{2}; \ldots; v_{m}\}$. Note that in this way every wildcard is mapped to a non-empty substring of $t$ and no two wildcards are mapped to the same substring of $t$, as required.

We then define the $k'$-GP-matching function $f$ as follows: $f(;) = \$, $f(\oplus) = \oplus$, $f(#) = \#$, $f(V_{i}) = v_{h_{i}}$, $f(D) = \%$. For every $i$ and $h_{i}$ with $1 \leq i \leq k$ and $1 \leq h_{i} \leq n$. It is straightforward to check that $f$ together with the mapping for the wildcards $k'$-GP-matches $p$ to $t$.

Lemma 3 Let $f$ be a function that $k'$-GF-matches $p$ to $t$, then: $f(;) = \$, $f(\oplus) = \oplus$, $f(#) = \#$, and $f(D) = \%$. Moreover, all wildcards have to be placed on all the $k'$ occurrences of $D$ in $p'$.

Proof We first show that $f(D) = \%$. Observe that the string $t'$ is square-free (recall the definition of square-free from Sect. 2). It follows that every two consecutive occurrences of pattern letters in $p''$ have to be mapped to a substring of $t''$. Because there are $2(k' + 1)$ occurrences of $D$ in $p''$ it follows that at least two consecutive occurrences of $D$ in $p''$ are not replaced with wildcards and hence $D$ has to be mapped to a substring of $t''$. Furthermore, since all occurrences of $D$ are at the end of $p''$, we obtain that $D$ has to be mapped to $\%$, as required. Because all occurrences of $D$ in $p'$ have to be mapped to substrings of $t'$ and $t'$ does not contain the letter $\%$, it follows that all the $k'$ occurrences of $D$ in $p'$ have to be replaced by wildcards. Since we are only allowed to use at most $k'$ wildcards, this shows the second statement of the lemma. Since no wildcards are used to replace letters in $p''$, it now easily follows that $f(;) = \$, $f(\oplus) = \oplus$ and $f(#) = \#$.

Lemma 4 If $p$ $k'$-GF/GP-matches to $t$, then $G$ has a $k$-clique.

Proof Because every GP-matching is also a GF-matching, it is sufficient to show the statement given a $k'$-GF-matching of $p$ to $t$.

Let $f$ be a function that $k'$-GF-matches $p$ to $t$. We start by showing that $|f(V_{i})| = 1$ for every $i$ with $1 \leq i \leq k$. Suppose for a contradiction that this is not the case, i.e., there is an $h$ with $1 \leq h \leq k$ such that $|f(V_{h})| > 1$. Because of Lemma 3, we know that $f(#) = \#$, $f(\oplus) = \oplus$ and $f(;) = \$, and that no occurrence of $\#$, $\oplus$ and ;, respectively, in $p$ is replaced by a wildcard. Since the number of occurrences of $\#$ in $t$ is equal to the number of occurrences of $\#$ in $p$, we obtain that the $i$th occurrence of $\#$ in $p$ is mapped to the $i$th occurrence of $\#$ in $t$. Consequently, for every $i$ and $j$
with \(1 \leq i < j \leq k\), we obtain that the substring \(\Sigma_i \cap \Sigma_j\) is mapped to a substring of the string \(l_{i,j}; v(t(e_{1,2}^i)); \ldots; v(t(e_{m,2}^i)); r_{i,j}\) in \(t\). Moreover, because \(f(\@) = \@\) and \(f(\cdot) = ;\), it follows that if \(|f(\Sigma_i)| > 1\) then the string \(f(\Sigma_i)\) contains at least one character \(v_i^l\) and at least one character \(v_j^l\) for some \(l\) and \(s\) with \(1 \leq l, s \leq n\) and the analogous property holds for \(\Sigma_j\). Consequently, if \(|f(\Sigma_i)| > 1\) then for every \(i\) with \(1 \leq i \leq k\), the string \(f(\Sigma_i)\) has to contain at least one character \(v_i^l\) for some \(l\) with \(1 \leq l \leq m\), contradicting the fact that for every \(i\) and \(j\) with \(1 \leq i < j \leq k\) the string \(\Sigma_i \cap \Sigma_j\) is mapped to a substring of \(l_{i,j}; v(t(e_{1,2}^i)); \ldots; v(t(e_{m,2}^i)); r_{i,j}\). This shows that \(|f(\Sigma_i)| = 1\) for every \(i\) with \(1 \leq i \leq k\).

We now claim that the set \(\{ f(\Sigma_i) \mid 1 \leq i \leq k \}\) is a \(k\)-clique of \(G\). Recall that from the above argumentation, we obtained that for every \(i\) and \(j\) with \(1 \leq i < j \leq k\) the substring \(\Sigma_i \cap \Sigma_j\) is mapped to a substring of the string \(l_{i,j}; v(t(e_{1,2}^i)); \ldots; v(t(e_{m,2}^i)); r_{i,j}\) in \(t\). Because \(f(\@) = \@\) and no occurrence of \(\@\) is replaced by a wildcard and furthermore \(|f(\Sigma_i)| = 1\) for every \(i\) with \(1 \leq i \leq k\), we obtain that both \(\Sigma_i\) and \(\Sigma_j\) are mapped to some letter \(v_i^l\) and \(v_j^l\) for some \(l\) and \(s\) with \(1 \leq l, s \leq n\) such that \(\{v_i^l, v_j^l\} \in E\). Hence, \(\{ f(\Sigma_i) \mid 1 \leq i \leq k \}\) is a \(k\)-clique of \(G\).

\(\square\)

Condition (C2) can now be obtained as follows: Lemma 2 shows the implication from S3 to S2, Lemma 4 shows the implication from S1 to S3, and the implication from S2 to S1 is trivially satisfied. This concludes the proof of Theorem 3.

**Theorem 4** Max-GFM and Max-GPM are W[1]-hard parameterized by \(|\Sigma_p|\), \(\max_i |f(p_i)|\), and \(\max |f(\cdot)|\) (even if \(\max_i |f(p_i)| \leq 1\) and \(\max |f(\cdot)| \leq 2\)).

Let \(q = \binom{k}{2}(8(m - 1))\). We will show the theorem by constructing a text \(t\) and a pattern \(p\) from \(G\) and \(k\) in polynomial time such that the following conditions hold:

(C1) the parameter \(|\Sigma_p|\) can be bounded by a function of \(k\).

(C2) The following statements are equivalent:

- \(S1\) \(q\)-GF/GP-matches \(t\);
- \(S2\) \(q\)-GF/GP-matches \(t\) using a function \(f\) with \(\max_{p \in \Sigma_p} |f(p)| \leq 1\) and \(\max |f(\cdot)| \leq 2\);
- \(S3\) \(G\) has a \(k\)-clique.

We set \(\Sigma_t = \{; , \@, \#, \Diamond, \%\} \cup \{v_i^l \mid 1 \leq i \leq n\} \cup \{l_{i,j}^l, r_{i,j}^l \mid 1 \leq i < j \leq k\} \cup \{i_{x}^{l}, r_{x}^{l} \mid 1 \leq x \leq 4(m - 1)\}\) and \(\Sigma_p = \{; , \@, \#, \Diamond\} \cup \{\Sigma_i \mid 1 \leq i \leq k\}\).

We first define a preliminary text \(t'\) as follows.

\[
\begin{align*}
#\text{enu}(l_{1,2}^{1,2}, 4(m - 1)) & ; v(t_{e_{1,2}^{1,2}}); \ldots; v(t_{e_{m,2}^{1,2}}); \text{enu}(r_{1,2}^{1,2}, 4(m - 1))
\end{align*}
\]

\[
\begin{align*}
#\text{enu}(l_{1,k}^{1,k}, 4(m - 1)) & ; v(t_{e_{1,k}^{1,k}}); \ldots; v(t_{e_{m,k}^{1,k}}); \text{enu}(r_{1,k}^{1,k}, 4(m - 1))
\end{align*}
\]

\[
\begin{align*}
#\text{enu}(l_{2,3}^{2,3}, 4(m - 1)) & ; v(t_{e_{2,3}^{2,3}}); \ldots; v(t_{e_{m,3}^{2,3}}); \text{enu}(r_{2,3}^{2,3}, 4(m - 1))
\end{align*}
\]

\[
\begin{align*}
#\text{enu}(l_{2,k}^{2,k}, 4(m - 1)) & ; v(t_{e_{2,k}^{2,k}}); \ldots; v(t_{e_{m,k}^{2,k}}); \text{enu}(r_{2,k}^{2,k}, 4(m - 1))
\end{align*}
\]

\[
\begin{align*}
#\text{enu}(l_{k-1,1}^{k-1,1}, 4(m - 1)) & ; v(t_{e_{1,1}^{k-1,1}}); \ldots; v(t_{e_{m,1}^{k-1,1}}); \text{enu}(r_{k-1,1}^{k-1,1}, 4(m - 1))
\end{align*}
\]
We also need to define a preliminary pattern $p'$ as follows.

\[
\begin{align*}
\# \text{rp}(\Diamond, 4(m - 1)); \ & V_1 @ V_2; \ & \text{rp}(\Diamond, 4(m - 1))
\# & \text{rp}(\Diamond, 4(m - 1)); \ & V_1 @ V_k; \ & \text{rp}(\Diamond, 4(m - 1))
\# & \text{rp}(\Diamond, 4(m - 1)); \ & V_2 @ V_3; \ & \text{rp}(\Diamond, 4(m - 1))
\# & \text{rp}(\Diamond, 4(m - 1)); \ & V_2 @ V_k; \ & \text{rp}(\Diamond, 4(m - 1))
\ldots
\# & \text{rp}(\Diamond, 4(m - 1)); \ & V_{k-1} @ V_k; \ & \text{rp}(\Diamond, 4(m - 1))
\end{align*}
\]

Observe that the construction is similar to the construction in the proof of Theorem 3 in that it is a vertex-type reduction. Hence, again the text $t'$ and the pattern $p'$ ensure that for every pair $i$ and $j$ with $1 \leq i < j \leq k$, it holds that the vertices corresponding to the letters mapped to $V_i$ and $V_j$ are the endpoints of an edge in $E_i, j$, which implies that the set of vertices corresponding to the letters mapped to $V_1, \ldots, V_k$ is a $k$-clique of $G$. However, this time both $\max_i |f(p_i)|$ and $\max |f(\?)|$ are bounded by the parameter, in fact $\max_i |f(p_i)| \leq 1$ and $\max |f(\?)| \leq 2$. It is hence not possible to place a single letter (in the previous construction we used the letter $D$ for that purpose) to the left and the right of the occurrences of $V_1, \ldots, V_k$. Instead of the single letter $D$, we use many occurrences of the letter $\Diamond$ that have to be replaced by wildcards, which are then in turn mapped to substrings of length either one or two. Adjusting the number of these wildcards that are mapped to substrings of length one or two, respectively, then allows the flexibility required to properly map the letters $V_1, \ldots, V_k$.

To ensure that the separator letters $\Diamond, ;, @, \text{and} \#$ of the pattern must be mapped to the corresponding separator letters of the text, which in the case of the letter $\Diamond$ also ensures that all occurrences of $\Diamond$ in $p'$ are replaced by wildcards, $t$ and $p$ are obtained after preceding $t'$ and $p'$ with the following text.

\[
\text{rp}(\Diamond, 2q + 1)\text{rp}(; , 2q + 1)\text{rp}(@, 2q + 1)\text{rp}(\#, 2q + 1)
\]

This completes the construction of $t$ and $p$. Clearly, $t$ and $p$ can be constructed from $G$ and $k$ in fpt-time (even polynomial time). Furthermore, $|\Sigma_p| = k + 4$ showing condition (C1). It remains to show condition (C2), for which we will need the following intermediate lemmas.

**Lemma 5** If $G$ has a $k$-clique then $p$ $q$-GF/GP-matches $t$ using a function $f$ with $\max_{p \in \Sigma_p} |f(p)| \leq 1$ and $\max |f(\?)| \leq 2$.

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show that $p$ $q$-GP-matches to $t$.

Let $\{v_{h_1}^i, \ldots, v_{h_k}^k\}$ be the vertices and $\{e_{h_i,j}^i; 1 \leq i < j \leq k\}$ be the edges of a $k$-clique of $G$ with $1 \leq h_j \leq n$ and $1 \leq h_i,j \leq m$ for every $i$ and $j$ with $1 \leq i < j \leq k$. The function $f$ that $r$-GP-matching $p$ to $t$ is defined as follows: $f(\Diamond) = \Diamond; f(;) = ;, f(@) = @, f(\#) = \#$, and $f(V_i) = v_{h_i}^i$ for every $i$ with $1 \leq i \leq k$.}

\copyright Springer
We put $q$ wildcards on the last $q$ occurrences of $\diamond$ in $p$, i.e., every occurrence of $\diamond$ in $p'$. The length of the text the wildcards are mapped to is determined as follows. For an edge $e^{i,j}_{h,i,j}$ consider the substring of $p'$ corresponding to $e^{i,j}_{h,i,j}$, i.e., the substring $\text{rp}(\diamond, (4m-1)) \ast V \ast \text{rp}((m-1))$. The first $4(h_{i,j} - 1)$ and the last $4(m-h_{i,j})$ occurrences of $\diamond$ (in this substring) are replaced with wildcards that are mapped to texts of length 2. All the remaining occurrences of $\diamond$ (in this substring) are replaced with wildcards that are mapped to texts of length 1. Note that because each of the wildcards is mapped to a substring containing a letter that occurs only once in $t$, i.e., one of the letters $l^{i,j}_x$ and $r^{i,j}_x$, the mapping of the wildcards is injective. Then, $\max_{p \in \Sigma} |f(p)| \leq 1$, $\max |f(\cdot)| \leq 2$, $f$ is injective and it is now straightforward to verify that $f$ $q$-GP-matches $p$ to $t$. \hfill \square

**Lemma 6** For any function $f$ that $q$-GF-matches $p$ to $t$ it holds that: $f(\diamond) = \diamond$, $f(\cdot) = \cdot$, $f(\circ) = \circ$, and $f(\#) = \#$.

**Proof** We first show that $|f(\diamond)| = |f(\cdot)| = |f(\circ)| = |f(\#)| = 1$. Suppose for the sake of contradiction that this does not hold, i.e., one of these letters, in the following denoted by $l$, is mapped to more than one letter in the text. Because $l$ appears at least $2q+1$ times in $p$, we obtain that at least $q+1$ occurrences of $l$ in $p$ are not replaced by a wildcard. However, it is easy to verify that $t'$ does not contain $q+1$ occurrences of any string of length at least 2. This shows that $|f(\diamond)| = |f(\cdot)| = |f(\circ)| = |f(\#)| = 1$. It remains to show that $f(\diamond) = \diamond$, $f(\cdot) = \cdot$, $f(\circ) = \circ$, and $f(\#) = \#$. Suppose for a contradiction that this is not the case. Then either:

- One of these letters, in the following denoted by $l$, is mapped to a letter in $\Sigma_t \setminus \{\diamond, \cdot, \circ, \#\}$. Because $l$ occurs at least $2q+1$ times in $p$, we obtain that at least $q+1$ occurrences of $l$ are not mapped to a wildcard. However, none of the letters in $\Sigma_t \setminus \{\diamond, \cdot, \circ, \#\}$ occurs more than $q$ times, contradicting our assumption that $f$ is a $q$-GF-matching from $p$ to $t$.

- Otherwise, there are at least two of these letters, in the following denoted by $l$ and $l'$, such that $f(l) = f(l') \in \{\diamond, \cdot, \circ, \#\}$. Together $l$ and $l'$ occur at least $4q+2$ times in $p$ and at least $3q+2$ of these occurrences are not replaced by wildcards. However, no letter in $\Sigma_t$ occurs more than $3q+1$ times, contradicting our assumption that $f$ is a $q$-GF-matching from $p$ to $t$.

Hence, in both of the above cases we obtain a contradiction to our assumption that $f$ is a $q$-GF-matching from $p$ to $t$. This completes the proof of the lemma. \hfill \square

**Lemma 7** Any $q$-GF-matching of $p$ to $t$ replaces all the $q$ occurrences of $\diamond$ in $p'$ with wildcards.

**Proof** It follows from the previous lemma that $f(\diamond) = \diamond$ for any function that $q$-GF-matches $p$ to $t$. Because $\diamond$ occurs exactly $3q+1$ times in $p$ but only $2q+1$ times in $t$, it follows that at least $q$ occurrences of $\diamond$ in $p$ have to be replaced by a wildcard. Since we are only allowed to replace at most $q$ letters of $p$ with a wildcard and all occurrences of $\diamond$ in $t$ are at the beginning of $t$, the claim of the lemma follows. \hfill \square

**Lemma 8** If $p$ $q$-GF/GP-matches $t$ then $G$ has a $k$-clique.
Proof Because every GP-matching is also a GF-matching, it is sufficient to show the
statement given a q-GF-matching of p to t.
Let f be a function that q-GF-matches p to t. Because of Lemma 6, it holds that
f(∅) = ∅, f(;) = ;, f(}@ = @, and f(#) = #. Furthermore, because of Lemma 7 the
only letters in p that are replaced with wildcards are the last q occurrences of ∅
in p. Because the number of occurrences of the letter # is the same in t and p each
occurrence of # in p has to be mapped to its corresponding occurrence in t. We obtain
that for every i and j with 1 ≤ i < j ≤ k the substring

\[ \text{rp}(∅, 4(m − 1)); \text{V}_i @ \text{V}_j; \text{rp}(∅, 4(m − 1)) \]

of p has to be mapped to the substring

\[ \text{enu}(l^{i,j}, 4(m − 1)); \text{vt}(e^{i,j}_1); \cdots; \text{vt}(e^{i,j}_m); \text{enu}(r^{i,j}, 4(m − 1)) \]

of t. Furthermore, because f(; ) = ; and f(}@ = @, we obtain that:

(*) For every i and j with 1 ≤ i < j ≤ k the substring \( \text{V}_i @ \text{V}_j \); has to be mapped
to a substring of \( \text{vt}(e^{i,j}_1); \cdots; \text{vt}(e^{i,j}_m) \).

We show next that for every i with 1 ≤ i ≤ k, \( f(\text{V}_i) = v^i_1 \) for some l with 1 ≤ l ≤ n.
Suppose not, and let j with 1 ≤ j ≤ k and j ̸= i. Then because of (*), \( f(\text{V}_i) \) contains
at least one letter \( v^i_l \) and at least one letter \( v^j_l \) for some l and s with 1 ≤ l, s ≤ n. Let \( j' \)
with 1 ≤ j' ≤ k such that \( j' ̸= i \) and \( j' ̸= j \). Because of (*) applied to the pair \( i \) and \( j' \),
we obtain that either \( \text{V}_i @ \text{V}_{j'} \); is mapped to the substring
\[ \text{vt}(e^{i,j'}_1); \cdots; \text{vt}(e^{i,j'}_m) \]; (if \( i < j' \)) or \( \text{V}_{j'} @ \text{V}_i \); is mapped to the substring
\[ \text{vt}(e^{j',i}_1); \cdots; \text{vt}(e^{j',i}_m) \); (if \( i > j' \)).

Hence, in both cases \( \text{V}_i \) is mapped to a substring that does not contain \( v^i_{i,l} \), contradicting
the fact that \( \text{V}_i \) is never replaced by a wildcard. This shows that for every i with
1 ≤ i ≤ k, \( f(\text{V}_i) = v^i_1 \) for some l with 1 ≤ l ≤ n and consequently for every i and
j with 1 ≤ i < j ≤ k, the substring \( \text{V}_i @ \text{V}_j \) of p has to be mapped to a substring
\( \text{vt}(e^{i,j}_1) \) of t for some 1 ≤ l ≤ m. Hence, the set \{ \( f(\text{V}_i) \) | 1 ≤ i ≤ k \} is a k-clique of
\( G \).

Condition (C2) can now be obtained as follows: Lemma 5 shows the implication from
S3 to S2, Lemma 8 shows the implication from S1 to S3, and the implication from S2
to S1 is trivially satisfied. This concludes the proof of Theorem 4.

The following theorems use an edge-type reduction. This is mainly because \#Σ_t
is part of the combined parameter, and hence we cannot repeat the text letters corre-
sponding to vertices of G arbitrary often anymore, making it basically impossible to
apply a vertex-type reduction.

**Theorem 5** GFM and GPM are \( \mathsf{W}[1] \)-hard parameterized by \#Σ_t, \#Σ_p, and \( |Σ_p| \).

Observe that the above theorem implies that also Max-GFM and Max-GPM are
\( \mathsf{W}[1] \)-hard additionally parameterized by \#? and \( \max |f(?)| \).

We will show the theorem by constructing a text t and a pattern p from G and k
in polynomial time such that p GF/GP-matches t if and only if G has a k-clique. The
alphabet \( Σ_t \) consists of:
– the letter # (used as a separator);
– one letter $a_e$ for every $e \in E$ (representing the edges of $G$);
– one letter $i$ for every $i$ with $1 \leq i \leq n$ (used as special separators that group edges incident to the same vertex);
– the letters $h_{i,j}$, $r_{i,j}$, $l_i$, $r_i$ for every $i$ and $j$ with $1 \leq i < j \leq k$ (used as dummy letters to ensure injectivity for GPM);
– the letter $d_v^i$ and $d_v^j$ for every $e \in E$ and $v \in V(G)$ with $v \in e$ (used as dummy letters to ensure injectivity for GPM).

We set $\Sigma_p = \{\#\} \cup \{E_{i,j}, L_{i,j}, R_{i,j}, L_i, R_i, A_i \mid 1 \leq i < j \leq k\} \cup \{D_{i,j} \mid 1 \leq i \leq k \text{ and } 1 \leq j \leq k \}$. Furthermore, for a vertex $v \in V(G)$, we write $e(v)$ to denote the text $el(v, E_1(v)) \cdots el(v, E_k(v))d_v$, where $el(v, E')$, for vertex $v$ and a set $E'$ of edges with $E' = \{e_1, \ldots, e_l\}$, is the text $d_{e_1}^v a_{e_1} d_{e_2}^v a_{e_2} \cdots d_{e_l}^v a_{e_l}$.

We first define the following preliminary text and pattern strings. Let $t_1$ be the text:

$$
\begin{align*}
#l_1 a_1^1 & \cdots a_{1,2} r_{1,2}^# & \cdots & #l_1 a_1^k & \cdots a_{1,k} r_{1,k} \\
#l_2 a_2^1 & \cdots a_{2,3} r_{2,3}^# & \cdots & #l_2 a_2^k & \cdots a_{2,k} r_{2,k} \\
& \cdots \\
#l_{k-1} a_{k-1,k} & \cdots a_{k,k} r_{k-1,k}
\end{align*}
$$

Let $t_2$ be the text:

$$
\begin{align*}
#l_1 #1 e(v_1^1) #1 & \cdots #n e(v_n^1) #1 r_1 \\
& \cdots \\
#l_k #1 e(v_1^k) #1 & \cdots #n e(v_n^k) #1 r_k 
\end{align*}
$$

Let $p_1$ be the pattern:

$$
\begin{align*}
#L_1 a_{1,2} R_{1,2} & \cdots #L_1 a_{1,k} R_{1,k} \\
#L_2 a_{2,3} R_{2,3} & \cdots #L_2 a_{2,k} R_{2,k} \\
& \cdots \\
#L_{k-1} a_{k-1,k} & \cdots a_{k,k} R_{k-1,k}
\end{align*}
$$

For $i, j$ with $1 \leq i, j \leq k$, let $I(i, j)$ be the letter $E_{i,j}$ if $i < j$, the letter $E_{j,i}$ if $i > j$ and the empty string if $i = j$. We define $p(1)$ to be the pattern:

$$A_1 D_{1,2} I(1, 2) D_{1,3} I(1, 3) \cdots D_{1,k} I(1, k) D_{1,k+1} A_1$$

we define $p(k)$ to be the pattern:

$$A_k D_{k,1} I(k, 1) D_{k,2} I(k, 2) \cdots D_{k,k-1} I(k, k-1) D_{k,k+1} A_k$$

and for every $i$ with $1 < i < k$, we define $p(i)$ to be the pattern:
\[ A_iD_{i,1}I(i, 1)D_{i,2}I(i, 2) \cdots D_{i,i-1}I(i, i - 1) \]
\[ D_{i,i+1}I(i, i + 1) \cdots D_{i,k}I(i, k)D_{i,k+1}A_i \]

Then \( p_2 \) is the pattern:

\[ \#L_1p(1)R_1\# \cdots \#Lkp(k)R_k\# \]

We also define \( t_0 \) to be the text \(#\#\) and \( p_0 \) to be the pattern \(#\#\). Then, \( t \) is the concatenation of \( t_0, t_1 \) and \( t_2 \) and \( p \) is a concatenation of \( p_0, p_1 \) and \( p_2 \).

Informally, the text \( t_0 \) and the pattern \( p_0 \) ensure that the pattern letter \( \# \) has to be mapped to the text letter \( \# \). Furthermore, the text \( t_1 \) and the pattern \( p_1 \) ensure that for every \( i \) and \( j \) with \( 1 \leq i < j \leq k \), the pattern letter \( E_{i,j} \) is mapped to a text letter \( a_e \), where \( e \in E_{i,j} \), and hence is mapped to a text corresponding to an edge in \( E_{i,j} \). Finally, the text \( t_2 \) and the pattern \( p_2 \) ensure that for every \( i \) with \( 1 \leq i \leq k \), all edges corresponding to the text mapped to any \( E_{i,j} \) with \( j \neq i \) have the same endpoint in \( V_i \). Together this means that the set of edges corresponding to the text mapped to all the letters \( E_{i,j} \) forms a \( k \)-clique of \( G \).

This completes the construction of \( t \) and \( p \). Clearly, \( t \) and \( p \) can be constructed from \( G \) and \( k \) in fpt-time (even polynomial time). Furthermore, \( \#\Sigma_t = \binom{k}{2} + k + 3 \), \( \#\Sigma_p = \binom{k}{2} + k + 3 \), \( |\Sigma_p| = k(k + 1) + 3\binom{k}{2} + 3k + 1 \) and hence bounded by \( k \), as required. It remains to show that \( G \) has a \( k \)-clique if and only if \( p \) GF/GP-matches \( t \).

**Lemma 9** If \( G \) has a \( k \)-clique then \( p \) GF/GP-matches \( t \).

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show that \( p \) \( k \)-GP-matches to \( t \).

Let \( \{v_{h_1}^1, \ldots, v_{h_k}^k\} \) be the vertices and \( \{e_{ij}^{i,j} \mid 1 \leq i < j \leq k\} \) be the edges of a \( k \)-clique of \( G \) with \( 1 \leq h_i \leq n \) and \( 1 \leq h_i, j \leq m \) for every \( i \) and \( j \) with \( 1 \leq i < j \leq k \). We define the function \( f \) that \( k \)-GP-matches \( p \) to \( t \) as follows: We set \( f(\#) = \# \). Moreover, for every \( i \) and \( j \) with \( 1 \leq i < j \leq k \), we set \( f(E_{i,j}) = a_{e_{ij}}, f(A_i) = \#i, f(L_{i,j}) = l_i j a_{e_j} \cdots a_{e_{h_i,j-1}}, f(R_{i,j}) = a_{e_{j,i}} \cdots a_{e_{h_i,j+1}} r_{i,j} \), \( f(L_i) = \#l_i \#1 e(v_1^i) \#1 \cdots \#h_{i-1} e(v_{h_{i-1}}^i) \#h_i \#1, \) and \( f(R_i) = \#h_{i-1+1} e(v_{h_{i-1}}^i) \#h_{i+1} \cdots \#n e(v_n^i) \#n r_i \).

For every \( i \) and \( j \) with \( i \neq j \), let \( e(i,j) \) be the edge \( e_{h_i,j}^{i,j} \) if \( i < j \) and the edge \( e_{h_j,i}^{j,i} \) otherwise. Then, the letters \( D_{i,j} \) are mapped as follows:

- For every \( i \) and \( j \) with \( 1 \leq i < k \), \( 2 \leq j \leq k \), \( i \neq j \), and \( (i, j) \neq (1, 2) \), we map \( f(D_{i,j}) \) to the substring of \( e(v_{h_j}^i) \) in between the occurrences (and not including these occurrences) of the letters \( e(i, j - 1) \) and \( e(i, j) \).
- We map \( f(D_{1,2}) \) to be the prefix of \( e(v_{h_1}^1) \) ending before the letter \( e(1, 2) \).
- For every \( i \) with \( 2 \leq i \leq k \), we map \( f(D_{i,1}) \) to the prefix of \( e(v_{h_1}^i) \) ending before the letter \( e(i, 1) \).
- For every \( i \) with \( 1 \leq i < k \), we map \( f(D_{i,k+1}) \) to the suffix of \( e(v_{h_i}^i) \) starting after the letter \( e(i, k) \).
We map $f(D_{k,k+1})$ to be the suffix of $e(v_{h_i}^1)$ starting after the letter $e(k, k - 1)$. Observe because $f$ maps each of the letters $L_{i,j}, R_{i,j}, L_i, R_i,$ and $D_{i,j}$ to a substring of $t$ containing a letter that occurs only once in $t$, i.e., the letters $h_{i,j}, r_{i,j}, l_i, r_i,$ and $d_{i,j}^u$, respectively, and the mappings for $\#$, $E_{i,j}$, and $A_i$ are obviously pairwise distinct, we obtain that $f$ is injective. It is now straightforward to check that $f$ GF-matches $p$ to $t$.

**Lemma 10** If $p$ GF/GP-matches $t$, then $G$ has a $k$-clique.

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show the statement given a GF-matching of $p$ to $t$.

Let $f'$ be a function that GF-matches $p$ to $t$. We first show that $f'(\#) = \#$. Suppose for a contradiction that $f'(\#) \neq \#$. Because $t$ and $p$ start with $$ it follows that $f'(\#)$ is a string that starts with ##. However, $t$ does not contain any other occurrence of the string $$ and hence the remaining occurrences of $\#$ in $p$ cannot be matched by $f$.

Because $t$ and $p$ have the same number of occurrences of $\#$, it follows that the $i$th occurrences of $\#$ in $p$ has to be mapped to the $i$th occurrence of $\#$ in $t$. We obtain that:

1. For every $i$, $j$ with $1 \leq i < j \leq k$, the substring $L_i, E_{i,j} R_i, j$ of $p$ has to be mapped to the substring $l_i, a_{i,j} \cdots a_{i,j} r_i, j$ of $t$.
2. For every $i$ with $1 \leq i \leq k$, the substring $L_i p(i) R_i$ of $p$ has to be mapped to the substring $l_i, \#_1 e(v_{i,1}^1) \#_1 \cdots \#_n e(v_{i,n}^1) \#_n r_i$ of $t$.

Because for every $i$ with $1 \leq i \leq k$ the letters $\#_j$ are the only letters that occur more than once in the substring $l_i, \#_1 e(v_{i,1}^1) \#_1 \cdots \#_n e(v_{i,n}^1) \#_n r_i$ of $t$, we obtain from (2) that $A_i$ has to be mapped to $\#_j$ for some $j$ with $1 \leq j \leq n$. Consequently:

3. For every $i$ with $1 \leq i \leq k$, the substring $p(i)$ of $p$ has to be mapped to a substring $\#_j e(v_{i,j}^1) \#_j$ of $t$ for some $j$ with $1 \leq j \leq n$.

It follows from (1) that for every $i$, $j$ with $1 \leq i < j \leq k$, $f(E_{i,j})$ is mapped to some edges between $V_i$ and $V_j$. We show next that $f$ maps $E_{i,j}$ to exactly one edge between $V_i$ and $V_j$. Assume not, then there are two edges mapped to $E_{i,j}$ via $f$ that either have different endpoints in $V_i$ or $V_j$. W.l.o.g. suppose that the former is the case, i.e., there are two edges $e$ and $e'$ contained in $f(E_{i,j})$ with distinct endpoints in $V_i$ and let $l$ with $1 \leq x \leq n$ be such that according to (3) the substring $p(i)$ of $p$ is mapped to the substring $\#_x e(v_{i,x}^1) \#_x$ of $t$. Because $\#_x e(v_{i,x}^1) \#_x$ contains only edges incident to the vertex $v_{i,x}$ in $V_i$ one of the two edges $e$ or $e'$ is not contained in $\#_x e(v_{i,x}^1) \#_x$ and hence the substring $p(i)$ (which contains $E_{i,j}$) cannot be mapped to $\#_x e(v_{i,x}^1) \#_x$ contradicting (3). This shows that $f$ maps $E_{i,j}$ to exactly one edge between $V_i$ and $V_j$. Furthermore, because of (3) it follows that for every $i$ with $1 \leq i \leq k$, it holds that the edges mapped to any $E_{x,y}$ with $1 \leq x < y \leq k$ such that $x = i$ or $y = i$ have the same endpoint in $V_j$. Hence, the set of edges mapped to the letters $E_{i,j}$ for $1 \leq i < j \leq k$ forms a $k$-clique of $G$.

This concludes the proof of Theorem 5.

**Theorem 6** Max-GFM and Max-GPM are $W[1]$-hard parameterized by $\# \Sigma_i$, $\# \Sigma_p$, $|\Sigma_p|$, $\max_i |f(p_i)|$, and $\#$ (even if $\max_i |f(p_i)| \leq 1$).

© Springer
Informally, the construction is very similar to the construction given in the proof of Theorem 5. The main difference is that now $\max_i |f(p_i)|$ is bounded by the parameter and hence we can no longer use single letters ($L_{i,j}, R_{i,j}, L_i, R_i, \text{and } D_{i,j}$) to shift the letters $E_{i,j}$ over the right position in the text. Instead we will introduce a letter $D$ and force it to be replaced by wildcards, which then can be freely mapped to substrings of the text of arbitrary length.

Let $k' = 2(\frac{\delta}{2}) + k(k + 2)$. We will show the theorem by constructing a text $t$ and a pattern $p$ from $G$ and $k$ in polynomial time such that the following two conditions are satisfied:

(C1) the parameters $\#\Sigma_l, \#\Sigma_p, |\Sigma_p|$, and $\#?$ can be bounded by a function of $k$.
(C2) The following statements are equivalent:

1. $p$ $k'$-GF/GP-matches $t$;
2. $p$ $k'$-GF/GP-matches $t$ using a function $f$ with $\max_{p \in \Sigma_p} |f(p)| \leq 1$;
3. $S_3$ has a $k$-clique.

The alphabet $\Sigma_t$ consists of:

- the letter $#$ (used as a separator);
- the letter $\%$ (used to force the wildcards);
- one letter $a_e$ for every $e \in E$ (representing the edges of $G$);
- one letter $#_i$ for every $i$ with $1 \leq i \leq n$ (used as separators that group edges incident to the same vertex);
- the letters $l_{i,j}, r_{i,j}, l_i, r_i$ for every $i$ and $j$ with $1 \leq i < j \leq k$ (used as dummy letters to ensure injectivity for GPM);
- the letter $d_v^e$ for every $e \in E$ and $v \in V(G)$ with $v \in e$ and the letter $d_v^v$ for every $v \in V(G)$ (used as dummy letters to ensure injectivity for GPM).

We set $\Sigma_p = \{#, D\} \cup \{E_{i,j} \mid 1 \leq i < j \leq k\}$.

For a vertex $v \in V(G)$, we write $e(v)$ to denote the text $el(v, E_1(v)) \cdot \cdot \cdot el(v, E_k(v))d_v^v$, where $el(v, E')$, for vertex $v$ and a set $E'$ of edges with $E' = \{e_1, \ldots, e_l\}$, is the text $d_{e_1}^v e_{e_1} d_{e_2}^v e_{e_2} \cdot \cdot \cdot d_{e_l}^v e_{e_l}$.

We first define the following preliminary text and pattern strings. Let $t_1$ be the text:

$$
\#l_{1,2}a_{e_{1,2}} \cdot \cdot \cdot a_{e_{m,2}}e_{1,2}r_{1,2}\# \cdot \cdot \cdot \#l_{1,k}a_{e_{1,k}} \cdot \cdot \cdot a_{e_{m,k}}e_{1,k}r_{1,k}$$
$$
\#l_{2,3}a_{e_{2,3}} \cdot \cdot \cdot a_{e_{m,3}}e_{2,3}r_{2,3}\# \cdot \cdot \cdot \#l_{2,k}a_{e_{2,k}} \cdot \cdot \cdot a_{e_{m,k}}e_{2,k}r_{2,k}$$
$$
\ldots$$
$$
\#l_{k-1,k}a_{e_{k-1,k}} \cdot \cdot \cdot a_{e_{m,k}}e_{k-1,k}r_{k-1,k}$$

Let $t_2$ be the text:

$$
\#l_1 \#e(v_1^1)\#_1 \cdot \cdot \cdot \#_n e(v_n^1)\#_n r_1$$

$$
\ldots$$
$$
\#l_k \#e(v_k^k)\#_1 \cdot \cdot \cdot \#_n e(v_n^k)\#_n r_k$$

\[ Springer \]
Let $p_1$ be the pattern:

$$\#D\mathcal{E}_{1,2}D\ldots\#D\mathcal{E}_{1,k}D$$
$$\#D\mathcal{E}_{2,3}D\ldots\#D\mathcal{E}_{2,k}D$$
$$\ldots$$
$$\#D\mathcal{E}_{k-1,k}D$$

For $i, j$ with $1 \leq i, j \leq k$, let $I(i, j)$ be the letter $\mathcal{E}_{i,j}$ if $i < j$, the letter $\mathcal{E}_{j,i}$ if $i > j$ and the empty string if $i = j$. We define $p(1)$ to be the pattern:

$$A_1DI(1, 2)DI(1, 3) \cdots DI(1, k)DA_1,$$

we define $p(k)$ to be the pattern:

$$A_kDI(k, 1)DI(k, 2) \cdots DI(k, k-1)DA_k,$$

and for every $i$ with $1 < i < k$, we define $p(i)$ to be the pattern:

$$A_iDI(i, 1)DI(i, 2) \cdots DI(i, i-1)DI(i, i+1) \cdots DI(i, k)DA_i$$

Then $p_2$ is the pattern:

$$\#L_1p(1)R_1\ldots\#L_kp(k)R_k\#$$

Let $q = 2(k' + 1)$. We define $t_0$ to be the text $\#rp(\%, q)$ and $p_0$ to be the pattern $\#rp(D, q)$. Then, $t$ is the concatenation of $t_0$, $t_1$ and $t_2$ and $p$ is a concatenation of $p_0$, $p_1$ and $p_2$.

As mentioned above the construction is very similar to the construction used in the previous theorem. In particular, the purposes of $t_1$ and $p_1$ and $t_2$ and $p_2$ are the same a before. Additionally, the text $t_0$ and the pattern $p_0$ ensure that the pattern letter $\#$ has to be mapped to the text letter $\#$ and that the pattern letter $D$ must be mapped to the text letter $\%$ and hence all occurrences of $D$ in $p_1$ and $p_2$ have to be mapped to wildcards.

This completes the construction of $t$ and $p$. Clearly, $t$ and $p$ can be constructed from $G$ and $k$ in fpt-time (even polynomial time). Furthermore, $\#\Sigma_t = q$, $\#\Sigma_p = q + k'$, $|\Sigma_p| = (\frac{k}{2}) + k + 2$, and $\#? = k'$, showing condition (C1). It remains to show condition C2, for which we need the following intermediate lemmas.

**Lemma 11** If $G$ has a $k$-clique then $p k'$-GF/GP-matches $t$ using a function $f$ with $\max_{p \in \Sigma_p} |f(p)| \leq 1$.

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show that $p k'$-GP-matches to $t$.

Let $\{v_{h_1}^1, \ldots, v_{h_k}^k\}$ be the vertices and $\{e_{h_i,j}^i \mid 1 \leq i < j \leq k\}$ be the edges of a $k$-clique of $G$ with $1 \leq h_i \leq n$ and $1 \leq h_i,j \leq m$ for every $i$ and $j$ with $1 \leq i < j \leq k$. 

$\square$ Springer
We define the function $f$ that $k'$-GP-matches $p$ to $t$ as follows: We set $f(\#) = \#$ and $f(D) = \%$. Moreover, for every $i$ and $j$ with $1 \leq i < j \leq k$, we set $f(E_{i,j}) = e_{i,j}$ and $f(A_i) = \#_i$. We put $k'$ wildcards on the last $k'$ occurrences of $D$ in $p$. The mapping of these wildcards is defined very similar to the mapping of the letters $L_i, j$, $R_i, j$, $L_i$, $R_i$, and $D_i, j$ in the proof of Lemma 9 and will not be repeated here. Using this mapping ensures that every wildcard is mapped to an non-empty substring of $t$ and no two wildcards are mapped to the same substring of $t$. It is straightforward to check that $f$ together with above mapping for the wildcards $k'$-GP-matches $p$ to $t$. □

**Lemma 12** Let $f$ be a function that $k'$-GF-matches $p$ to $t$, then: $f(\#) = \#$ and $f(D) = \%$. Moreover, all wildcards have to be placed on all the $k'$ occurrences of $D$ in $p_0$.

**Proof** We first show that $f(D) = \%$. Observe that the only squares in the string $t$ are contained in $t_0$ (recall the definition of squares from Sect. 2). It follows that every two consecutive occurrences of pattern letters in $p_0$ (which are not replaced by wildcards) have to be mapped to a substring of $t_0$. Because there are $2(k' + 1)$ occurrences of $D$ in $p_0$ it follows that at least two consecutive occurrences of $D$ in $p_0$ are not replaced with wildcards and hence $D$ has to be mapped to a substring of $t_0$. Furthermore, since all occurrences of $D$ are at the end of $p_0$, we obtain that $D$ has to be mapped to $\%$, as required. Because all occurrences of $D$ in $p_0$ have to be mapped to substrings of the concatenation of $t_1$ and $r_2$, but these strings do not contain the letter $\%$, it follows that all the $k'$ occurrences of $D$ in $p_1$ and $p_2$ have to be replaced by wildcards. Since we are only allowed to use at most $k'$ wildcards, this shows the second statement of the claim. Since no wildcards are used to replace letters in $p_0$ it now easily follows that $f(\#) = \#$. □

**Lemma 13** If $p$ $k'$-GF/GP-matches $t$, then $G$ has a $k$-clique.

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show the statement given a $k'$-GF-matching of $p$ to $t$.

Let $f$ be a function that $k'$-GF-matches $p$ to $t$. Because of Lemma 12, we know that $f(\#) = \#$ and that no occurrence of $\#$ in $p$ is replaced by a wildcard. Because $t$ and $p$ have the same number of occurrences of $\#$, it follows that the $i$th occurrences of $\#$ in $p$ has to be mapped to the $i$th occurrence of $\#$ in $t$. We obtain that:

1. For every $i, j$ with $1 \leq i < j \leq k$, the substring $DE_{i,j}D$ of $p$ has to be mapped to the substring $l_{i,j}a_{i,j}\cdots a_{i,j}r_{i,j}$ of $t$.
2. For every $i$ with $1 \leq i \leq k$, the substring $L_i p(i) R_i$ of $p$ has to be mapped to the substring $l_i \#_1 e(v'_1) \#_1 \cdots \#_n e(v'_n) \#_n r_i$ of $t$.

Because for every $i$ with $1 \leq i \leq k$ the letters $\#_j$ are the only letters that occur more than once in the substring $l_i \#_1 e(v'_1) \#_1 \cdots \#_n e(v'_n) \#_n r_i$ of $t$, we obtain from (2) that $A_i$ has to be mapped to $\#_j$ for some $j$ with $1 \leq j \leq n$. Consequently:

3. for every $i$ with $1 \leq i \leq k$, the substring $p(i)$ of $p$ has to be mapped to a substring $\#_j e(v'_j) \#_j$ of $t$ for some $j$ with $1 \leq j \leq n$. 

© Springer
It follows from (1) that for every \(i, j\) with \(1 \leq i < j \leq k\), \(f(E_{i,j})\) is mapped to an edge between \(V_i\) and \(V_j\). Furthermore, because of (3) it follows that for every \(i\) with \(1 \leq i \leq k\), it holds that the edges mapped to any \(E_{x,y}\) with \(1 \leq x < y \leq k\) such that \(x = i\) or \(y = i\) have the same endpoint in \(V_i\). Hence, the set of edges mapped to the letters \(E_{i,j}\) for \(1 \leq i < j \leq k\) form a \(k\)-clique of \(G\).

\[\square\]

Condition (C2) can now be obtained as follows: Lemma 11 shows the implication from S3 to S2, Lemma 13 shows the implication from S1 to S3, and the implication from S2 to S1 is trivially satisfied. This concludes the proof of Theorem 6.

**Theorem 7** GFM and GPM are \(W[1]\)-hard parameterized by \(#\Sigma_t\), \(#\Sigma_p\), and \(\max_i |f(p_i)|\) (even if \(\max_i |f(p_i)| \leq 2\)).

Observe that the above theorem implies \(W[1]\)-hardness for Max-GFM and Max-GPM additionally parameterized by \(#\Sigma_t\), and \(\max_i |f(p)|\) (even if \(\max_i |f(p)| = 0\)).

Informally, the construction is very similar to the construction used in the proof of Theorem 5. The main difference is that because \(\max_i |f(p_i)|\) is bounded by the parameter, it is not sufficient anymore to use single “dummy” letters to the left and right of the letters \(E_{i,j}\). Instead, we need to use many distinct “dummy” letters—these “dummy” letters need to be distinct because also \(#\Sigma_p\) is bounded by the parameter—to the left and right of \(E_{i,j}\) each of which can be assigned either to a substring of length one or two. Similar to the construction used in the proof of Theorem 4, this allows us to freely map the letters \(E_{i,j}\) by varying the number of “dummy” letters that are mapped to a substring of length one or two, respectively.

We will show the theorem by constructing a text \(t\) and a pattern \(p\) from \(G\) and \(k\) such that the following two conditions hold:

(C1) the parameters \(#\Sigma_t\) and \(#\Sigma_p\) are bounded by \(k\).

(C2) The following statements are equivalent:

- S1 \(p\) GF/GP-matches \(t\);
- S2 \(p\) GF/GP-matches \(t\) using a function \(f\) with \(\max_{p \in \Sigma_p} |f(p)| \leq 2\);
- S3 \(G\) has a \(k\)-clique.

Let \(q = 2kn(n - 1) + 2n + (k - 1)m - 1\). The alphabet \(\Sigma_t\) consists of:

- the letter \# (used as a separator);
- the letter \# for every \(1 \leq i \leq n\) (used as a separator);
- the letters \(l_{i,j}^x\) and \(r_{i,j}^x\) for every \(1 \leq i < j \leq k\) and \(1 \leq x \leq m - 1\) (used as dummy letters allowing to choose an edge between \(V_i\) and \(V_j\));
- the letters \(l_{v,j}^x\) and \(r_{v,j}^x\) for every \(v \in V_i\), \(1 \leq j \leq k\), and \(1 \leq x \leq n - 1\), where \(1 \leq i \leq k\) and \(j \neq i\) (used as dummy letters allowing to an endpoint of an edge incident to \(v\) in \(V_j\));
- the letters \(l_i^x\) and \(r_i^x\) for every \(1 \leq i \leq k\) and \(1 \leq x \leq q\) (used as dummy letters allowing to choose a vertex in \(V_i\));
- the letter \(e_{i,j}^x\) for every \(1 \leq i < j \leq k\) and \(1 \leq x \leq m\) (representing the edges of \(G\)).

The alphabet \(\Sigma_p\) consists of:
more, we denote by

\[ t_{1,2} \]

if \( i < j \leq k \) and \( 1 \leq x \leq m - 1 \) (used as placeholders allowing to choose an edge between \( V_i \) and \( V_j \));

- the letters \( L_{x,y}^{i,j} \) and \( R_{x,y}^{i,j} \) for every \( 1 \leq i, j \leq k \) with \( i \neq j \), and \( 1 \leq x \leq n - 1 \) (used as placeholders allowing to verify an edge between \( V_i \) and \( V_j \));

- the letters \( L_{x}^{i,j} \) and \( R_{x}^{i,j} \) for every \( 1 \leq i, j \leq k \) and \( 1 \leq x \leq q \) (used as placeholders allowing to verify that the edges chosen incident to \( V_i \) have the same endpoint);

- the letter \( E_{i,j} \) for every \( 1 \leq i < j \leq k \) (holds the chosen edge between \( V_i \) and \( V_j \));

- the letter \( A_i \) for every \( 1 \leq i \leq n \) (used as separators).

Furthermore, for a vertex \( v \in V(G) \) and \( i \) with \( 1 \leq i \leq k \), we write \( e(v, i) \) to denote the text \( \text{el}(E_i(v)) \), where \( \text{el}(E') \) (for a set of edges \( E' \)) is a list of all the letters in \( \Sigma_t \) that correspond to the edges in \( E' \).

We first define the following preliminary text and pattern strings. For \( i \) and \( j \) with \( 1 \leq i < j \leq k \), we denote by \( t(i, j) \) the text:

\[
\text{enu}(l^{i,j}, m - 1)\text{enu}(e^{i,j}, m)\text{enu}(r^{i,j}, m - 1).
\]

Then, \( t_{1} \) is the text:

\[
\#t(1,2)\#\cdots\#t(1,k)\\
\#t(2,3)\#\cdots\#t(2,k)\\
\cdots\\
\#t(k - 1,k)\#
\]

For a vertex \( v \in V_i \), and \( j \) with \( 1 \leq j \leq k \), we denote by \( t(v, j) \) the text

\[
\text{enu}(l^{i,j}, n - 1)e(v, j)\text{enu}(r^{i,j}, n - 1)
\]

if \( j \neq i \) and the empty text if \( j = i \). Furthermore, we denote by \( t(v) \) the text \( t(v, 1) \cdots t(v, k) \). Let \( t_{2} \) be the text:

\[
\text{enu}(l^{1}, q)\#t(v_{1}^{1})\#\cdots\#t(v_{n}^{1})\#n\text{enu}(r^{1}, q)\\
\#\text{enu}(l^{2}, q)\#t(v_{1}^{2})\#\cdots\#t(v_{n}^{2})\#n\text{enu}(r^{2}, q)\\
\cdots\\
\#\text{enu}(l^{k}, q)\#t(v_{1}^{k})\#\cdots\#t(v_{n}^{k})\#n\text{enu}(r^{k}, q)
\]

For \( i \) and \( j \) with \( 1 \leq i < j \leq k \), we denote by \( p(i, j) \) the pattern

\[
\text{enu}(L^{i,j}, m - 1)E_{i,j}\text{enu}(R^{i,j}, m - 1).
\]

Let \( p_{1} \) be the pattern:

\[
\#p(1,2)\#\cdots\#p(1,k)\\
\#p(2,3)\#\cdots\#p(2,k)\\
\cdots\\
\#p(k - 1,k)\#
\]

For \( i, j \) with \( 1 \leq i, j \leq k \), let \( I(i, j) \) be the letter \( E_{i,j} \) if \( i < j \), the letter \( E_{j,i} \) if \( i > j \) and the empty string if \( i = j \). Furthermore, let \( p_{2} \) be the pattern

\[
\text{enu}(L^{i,j}, n - 1)I(i,j)\text{enu}(R^{i,j}, n - 1)
\]

if \( i \neq j \) and the empty pattern otherwise. Let
If $G$ has a k-clique then $p$ GF/GP-matches $t$ using a function $f$ with Lemma 14.

Because every GP-matching is also a GF-matching, it is sufficient to show that the role of the texts $t_0$, $t_1$, and $t_2$ and the patterns $p_0$, $p_1$, $p_2$ is very similar to the role they played in the proof of Theorem 5. That is the text $t_0$ and the pattern $p_0$ ensure that the pattern letter $#$ has to be mapped to the text letter $#$. Furthermore, the text $t_1$ and the pattern $p_1$ ensure that for every $i$ and $j$ with $1 \leq i < j \leq k$, the pattern letter $\mathcal{E}_{i,j}$ is mapped to a text letter $\mathcal{A}_e$, where $e \in \mathcal{E}_{i,j}$, and hence is mapped to a text corresponding to an edge in $\mathcal{E}_{i,j}$. Finally, the text $t_2$ and the pattern $p_2$ ensure that for every $i$ with $1 \leq i \leq k$, all edges corresponding to the text mapped to any $\mathcal{E}_{i,j}$ with $j \neq i$ have the same endpoint in $\mathcal{V}_i$. Together this means that the set of edges corresponding to the text mapped to the letters $\mathcal{E}_{i,j}$ forms a k-clique of $G$.

This completes the construction of $t$ and $p$. Clearly, $t$ and $p$ can be constructed from $G$ and $k$ in fpt-time (even polynomial time). Furthermore, because $\# \mathcal{S}_t = \binom{k}{2} + k + 2$, $\# \mathcal{S}_p = \binom{k}{2} + k + 2$, condition (C1) is satisfied. To show the remaining condition (C2) we need the following intermediate lemmas.

**Lemma 14** If $G$ has a k-clique then $p$ GF/GP-matches $t$ using a function $f$ with $\max_{p \in \mathcal{S}_p} |f(p)| \leq 2$.

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show that $p$ GP-matches to $t$.

Let $\{v^1_{h^1}, \ldots, v^k_{h^k}\}$ be the vertices and $\{e^i_{h,i,j} \mid 1 \leq i < j \leq k\}$ be the edges of a k-clique of $G$ with $1 \leq h_j \leq n$ and $1 \leq h_{i,j} \leq m$ for every $i$ and $j$ with $1 \leq i < j \leq k$.

We first give the GP-matching function $f$ for the letters in $\mathcal{S}_p$ that occur more than once in $p$ as follows: We set $f(\#) = \#$, $f(\mathcal{E}_{i,j}) = e^i_{h,i,j}$, and $f(\mathcal{A}_i) = h_i$, for every $i$ and $j$ with $1 \leq i < j \leq k$. Informally, we will map the remaining letters in $\mathcal{S}_p$ to substrings of $t$ of length between 1 and 2 in such a way that the occurrences of the letters $\#$, $\mathcal{E}_{i,j}$, and $\mathcal{A}_i$ are placed over the right positions in the text $t$. More formally, we define $f$ for the remaining letters in $\mathcal{S}_p$ as follows:

- For every $1 \leq i < j \leq k$, we define $f(L^{i,j}_x)$ in such a way that $|f(L^{i,j}_x)| = 2$ for every $1 \leq x \leq h_{i,j} - 1$ and $|f(L^{i,j}_x)| = 1$ for every $h_{i,j} - 1 < x \leq m - 1$.
- For every $1 \leq i < j \leq k$, we define $f(R^{i,j}_x)$ in such a way that $|f(R^{i,j}_x)| = 1$ for every $1 \leq x \leq h_{i,j} + 1$ and $|f(L^{i,j}_x)| = 2$ for every $h_{i,j} + 1 < x \leq m - 1$.
- For every $1 \leq i$, $j \leq k$ with $i \neq j$, we define $f(L^{i,j}_x)$ in such a way that $f(L^{i,j}_x) = 2$ for every $1 \leq x \leq s - 1$, where $s$ is the position of $e^i_{h,i,j}$ in $t(v_{hi}, j)$ and $f(L^{i,j}_x) = 1$ for every $s < x \leq n - 1$. 

\[ Springer \]
- For every $1 \leq i, j \leq k$ with $i \neq j$, we define $f(R^{i,j}_x)$ in such a way that $f(R^{i,j}_x) = 1$ for every $1 \leq x \leq s + 1$, where $s$ is the position of $e^{i,j}_{hi,j}$ in $t(v_{hi,j})$ and $f(R^{i,j}_x) = 1$ for every $s + 1 < x \leq n - 1$.
- For every $1 \leq i \leq k$, we define $f(L^i_x)$ in such a way that $|f(L^i_x)| = 2$ for every $1 \leq l \leq s - 1$, where $s$ is position of $h_{li}$ in the substring $#_{i}t(v_{i}^1)\#_{i}t(v_{i}^n)\#_n$ of $t$ and $|f(L^i_x)| = 1$ for every $s < x \leq q$.
- For every $1 \leq i \leq k$, we define $f(R^i_x)$ in such a way that $|f(R^i_x)| = 1$ for every $1 \leq x \leq s + 1$, where $s$ is position of $h_{li}$ in the substring $#_{i}t(v_{i}^1)\#_{i}t(v_{i}^n)\#_n$ of $t$ and $|f(R^i_x)| = 2$ for every $s + 1 < x \leq q$.

Observe because $f$ maps each of the letters $L^i_x$, $R^i_x$, $L^i_x$, $R^i_x$, $R^i_x$, and $R^i_x$ to a substring of $t$ containing a letter that occurs only once in $t$, i.e., the letters $l^i_x$, $r^i_x$, $l^i_x$, $r^i_x$, $l^i_x$, and $r^i_x$, respectively, and the mappings for $\#$, $\varepsilon_{i,j}$, and $A_{i}$ are obviously pairwise distinct, we obtain that $f$ is injective. It is now straightforward to check that $f$ GP-matches $p$ to $t$ and $\max_{p \in \Sigma} |f(p)| \leq 2$, as required.

**Lemma 15** If $p$ GF/GP-matches $t$, then $G$ has a $k$-clique.

**Proof** Because every GP-matching is also a GF-matching, it is sufficient to show the statement given a GF-matching of $p$ to $t$.

Let $f$ be the function that GF-matches $p$ to $t$. We first show that $f(\#) = \#$. Suppose for a contradiction that $f(\#) \neq \#$. Because $t$ and $p$ start with ## it follows that $f(\#)$ is a string that starts with ##. However, $t$ does not contain any other occurrence of the string ## and hence the remaining occurrences of $\#$ in $p$ cannot be matched by $f$.

Because $t$ and $p$ have the same number of occurrences of $\#$, it follows that the $i$th occurrences of $\#$ in $p$ has to be mapped to the $i$th occurrence of $\#$ in $t$. We obtain that:

1. for every $i, j$ with $1 \leq i < j \leq k$, the substring $p(i, j)$ of $p$ has to be mapped to the substring $t(i, j)$ of $t$.
2. for every $i$ with $1 \leq i \leq k$, the substring:

\[
\text{en}(\varepsilon^i, q)A_{i}p(e, 1)\cdots pe(i, k)A_{i}en(r^i, q)
\]

of $p$ has to be mapped to the substring:

\[
\text{en}(l^i, q)\#_{1}t(v_{i}^1)\#_{1}\cdots \#_{n}t(v_{i}^n)\#_{n}en(r^i, q)
\]

of $t$.

Because for every $i$ with $1 \leq i \leq k$ the letters $\#_{j}$ are the only letters that occur more than once in the substring $\text{en}(l^i, q)\#_{1}t(v_{i}^1)\#_{1}\cdots \#_{n}t(v_{i}^n)\#_{n}en(r^i, q)$ of $t$, we obtain that $A_{i}$ has to be mapped to $\#_{j}$ for some $j$ with $1 \leq j \leq n$. Consequently:

3. for every $i$ with $1 \leq i \leq k$, the substring $A_{i}pe(i, 1)\cdots pe(i, k)A_{i}$ of $p$ has to be mapped to a substring $\#_{j}t(v_{j}^1)\#_{j}$ of $t$ for some $j$ with $1 \leq j \leq n$.

It follows from (1) that for every $i, j$ with $1 \leq i < j \leq k$, the function $f$ maps $\varepsilon_{i,j}$ to edges between $V_{i}$ and $V_{j}$. We show next that $f$ maps $\varepsilon_{i,j}$ to exactly one edge
between \(V_i\) and \(V_j\). Assume not, then there are two edges mapped to \(E_{i,j}\) via \(f\) that have different endpoints in \(V_i\) or \(V_j\). W.l.o.g. suppose that the former is the case, i.e., there are two edges \(e \) and \(e'\) contained in \(f(E_{i,j})\) with distinct endpoints in \(V_i\) and let \(x\) with \(1 \leq x \leq n\) be such that according to (3) the substring \(A_i pe(i, 1) \cdots pe(i, k) A_i\) of \(p\) is mapped to the substring \(#_x t(v^i_1) \#_x\) of \(t\). Because \(#_x t(v^i_1) \#_x\) contains only edges incident to the vertex \(v^i_1\) in \(V_i\) one of the two edges \(e\) or \(e'\) is not contained in \(#_x t(v^i_1) \#_x\) and hence the substring \(A_j pe(i, 1) \cdots pe(i, k) A_i\) (which contains \(E_{i,j}\)) cannot be mapped to \(#_x t(v^j_1) \#_x\) contradicting (3). This shows that \(f\) maps \(E_{i,j}\) to exactly one edge between \(V_i\) and \(V_j\). Because of (3) it follows that for every \(i\) with \(1 \leq i \leq k\), it holds that the edges mapped to any \(E_{y,z}\) with \(1 \leq y < z \leq k\) such that \(y = i\) or \(z = i\) have the same endpoint in \(V_i\). Hence, the set of edges mapped to all the letters \(E_{i,j}\) for \(1 \leq i < j \leq k\) form a \(k\)-clique of \(G\). 

Condition (C2) can now be obtained as follows: Lemma 14 shows the implication from S3 to S2, Lemma 15 shows the implication from S1 to S3, and the implication from S2 to S1 is trivially satisfied. This concludes the proof of Theorem 7.

Acknowledgments  Sebastian Ordyniak acknowledges support from the Employment of Newly Graduated Doctors of Science for Scientific Excellence (CZ.1.07/2.3.00/30.0009).

References

1. Amir, A., Nor, I.: Generalized function matching. J. Discrete Algorithms 5(3), 514–523 (2007)
2. Angluin, D.: Finding patterns common to a set of strings (extended abstract). In: Proceedings of the 11th Annual ACM Symposium on Theory of Computing, April 30-May 2, 1979, Atlanta, GA, USA, pp. 130–141 (1979)
3. Angluin, D.: Finding patterns common to a set of strings. J. Comput. Syst. Sci. 21(1), 46–62 (1980)
4. Clifford, R., Harrow, A.W., Popa, A., Sach, B.: Generalised matching. In: Karlgren, J., Tarhio, J., Hyvöri, H. (eds.) Proceedings of the 16th International Symposium String Processing and Information Retrieval (SPIRE 2009), Saariselkä, Finland, August 25–27, 2009, Volume 5721 of Lecture Notes in Computer Science, pp. 295–301. Springer, Berlin (2009)
5. Clifford, R., Popa, A.: (In)approximability results for pattern matching problems. In: Holub, J., Zdárek, J. (eds.) Proceedings of the Prague Stringology Conference 2010, Prague, Czech Republic, August 30–September 1, 2010, pp. 52–62. Prague Stringology Club, Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague (2010)
6. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
7. Ehrenfreucht, A., Rozenberg, G.: Finding a homomorphism between two words in np-complete. Inf. Process. Lett. 9(2), 86–88 (1979)
8. Fernau, H., Schmid, M.L.: Pattern matching with variables: a multivariate complexity analysis. In: Fischer, J., Sanders, P. (eds.) Combinatorial pattern matching, Proceedings of 24th Annual Symposium, CPM 2013, Bad Herrenalb, Germany, June 17–19, 2013, Lecture Notes in Computer Science, vol. 7922, pp. 83–94. Springer, Berlin, Heidelberg (2013)
9. Fernau, H., Schmid, M.L., Villanger, Y.: On the parameterised complexity of string morphism problems. In: Seth, A., Vishnoi, N.K. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013), December 12–14, 2013, Guwahati, India, Volume 24 of LIPIcs, pp. 55–66. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik (2013)
10. Flum, J., Grohe, M.: Parameterized Complexity Theory, volume XIV of Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
11. Freydenberger, D.D., Reidenbach, D., Schneider, J.C.: Unambiguous morphic images of strings. Int. J. Found. Comput. Sci. 17(3), 601–628 (2006)
12. Jiang, T., Kinber, E., Salomaa, A., Salomaa, K., Sheng, Y.: Pattern languages with and without erasing. Int. J. Comput. Math. 50(3–4), 147–163 (1994)
13. Mateescu, A., Salomaa, A.: Finite degrees of ambiguity in pattern languages. Informatique Théorique et Applications 28(3–4), 233–253 (1994)

14. Ng, Y.K., Shinohara, T.: Developments from enquiries into the learnability of the pattern languages from positive data. Theor. Comput. Sci. 397(13), 150–165 (2008). Forty Years of Inductive Inference: Dedicated to the 60th Birthday of Rolf Wiehagen

15. Pietrzak, K.: On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. J. Comput. Syst. Sci. 67, 757–771 (2003)

16. Reidenbach, D.: A non-learnable class of e-pattern languages. Theor. Comput. Sci. 350(1), 91–102 (2006)

17. Reidenbach, D.: Discontinuities in pattern inference. Theor. Comput. Sci. 397(13), 166–193 (2008). Forty Years of Inductive Inference: Dedicated to the 60th Birthday of Rolf Wiehagen

18. Reidenbach, D., Schmid, M.L.: Patterns with bounded treewidth. Inf. Comput. 239, 87–99 (2014)

19. Schmid, M.L.: A note on the complexity of matching patterns with variables. Inf. Process. Lett. 113(19–21), 729–733 (2013)

20. Shinohara, T.: Polynomial Time Inference Of Extended Regular Pattern Languages. In: Goto, E., Furukawa, K., Nakajima, R., Nakata, I., Yonezawa, A. (eds.) RIMS Symposia on Software Science and Engineering, Volume 147 of Lecture Notes in Computer Science, pp. 115–127. Springer, Berlin (1983)