A Filtering Technique for All Pairs Approximate Parameterized String Matching

Shibsankar Das(✉)

Department of Mathematics, Institute of Science,
Banaras Hindu University, Varanasi 221005, Uttar Pradesh, India
shibsankar@bhu.ac.in, shib.iitm@gmail.com

Abstract. The paper deals with all pairs approximate parameterized string matching problem with error threshold k, among two sets of equal length strings. Let $P = \{p_1, p_2, \ldots, p_{n_P}\} \subseteq \Sigma_P^m$ and $T = \{t_1, t_2, \ldots, t_{n_T}\} \subseteq \Sigma_T^m$ be two sets of strings where $|\Sigma_P| = |\Sigma_T|$. For each $p_i \in P$, the problem is to find $t_j \in T$ which is approximately parameterized closest to p_i under the threshold. The solution has complexity $O(n_P\, n_T\, m)$. We introduce Parikh vector filtering technique in order to preprocess the given strings and avoid the unwanted paired comparisons. The PV-filtering does not change the asymptotic time complexity but rapidly improves running time for small error threshold as shown by experiments.

Keywords: Approximate parameterized string matching
Hamming distance · $\gamma(k)$-match of vectors · Parikh vector
PV-filtering technique

1 Introduction

The problem of searching a given string in a text has a wide range of applications such as in text-editing programs, search engines and searching for patterns in a DNA sequence. There are non-indexed and indexed versions of this problem. In the indexed version, it is allowed to preprocess the string (pattern or text) before searching for the pattern in the text. The motivation of preprocessing is to improve the efficiency of the search. The standard variations of string matching problems are exact string matching [7,14], parameterized string matching [3–5], approximate string matching [23] and approximate parameterized string matching [8,26].

A preliminary version of this work is submitted as a technical report in Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague [11]. Shibsankar Das was supported by the fellowship of HERITAGE Erasmus Mundus Partnership project for Ph.D. exchange mobility.

© Springer Nature Singapore Pte Ltd. 2018
D. Ghosh et al. (Eds.): ICMC 2018, CCIS 834, pp. 97–109, 2018.
https://doi.org/10.1007/978-981-13-0023-3_10

The Approximate Parameterized String Matching (APSM) problem is a well studied problem [1,2,13,19,21,25]. In [19], Hazay et al. have given reduction between the maximum weighted bipartite matching problem [12,15–17,20] and APSM problem for two equal length strings. They have used the maximum weighted bipartite (decomposition) algorithm, originally proposed by Kao et al. [20], to solve the APSM problem between two equal length strings $p \in \Sigma_P^*$ and $t \in \Sigma_T^*$ in time $O(m^{1.5})$, where $|p| = |t| = m$.

In this paper, we investigate All Pairs (best) Approximate Parameterized String Matching (APAPSM) problem with error threshold k (with respect to Hamming distance error model) among two sets of equal length strings. Let $P = \{p_1,\ p_2, \ldots, p_{n_P}\} \subseteq \Sigma_P^m$ and $T = \{t_1,\ t_2, \ldots, t_{n_T}\} \subseteq \Sigma_T^m$ be two sets of strings where $1 \leq i \leq n_P,\ 1 \leq j \leq n_T$ and $|\Sigma_P| = |\Sigma_T| = \sigma$. The APAPSM problem is to find: for each $p_i \in P$, a string $t_j \in T$ which is approximately parameterized closest to p_i under k threshold.

Section 2 describes the required preliminaries to understand the APAPSM problem which is explained in detail in the next section. In Sect. 3, we discuss a solution to the APAPSM problem with worst-case complexity $O(n_P\, n_T\, m)$, assuming a constant size alphabet. Next, we design a filtering technique by using Parikh vector [24] in order to preprocess the given strings and reduce the number of pair comparisons for solving APSM between the pair of strings with k error threshold. We call it PV-filter. Even though the filter does not improve the asymptotic bound theoretically, practical results in Sect. 4 show that it performs well for small error threshold. Finally, Sect. 5 summarizes the results.

2 Preliminaries and Related Results

We use some basic notions throughout the paper. An alphabet is a non-empty finite set of symbols. A *string* over a given alphabet is a finite sequence of symbols. We denote Σ^* as the set of all finite-length strings over alphabet Σ. The empty string is denoted by ε. The length of any string w is the total number of symbols in w and is denoted by $|w|$; so $|\varepsilon| = 0$. Let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ and for a given $m \in \mathbb{N}_0$, Σ^m is the set of all strings of length m over the alphabet Σ [26].

Let $w = xyz$ be a string where $x, y, z \in \Sigma^*$. We call y as a *substring* of string w. If $x = \varepsilon$ then y is a *prefix* of w. If $z = \varepsilon$ then y is a *suffix* of w. The i-th symbol of a string w is denoted by $w[i]$ for $1 \leq i \leq |w|$. We denote substring y of string w as $w[i..j]$ if y starts at position i and ends at position j for $1 \leq i \leq j \leq |w|$, and string $w[i..j] = \varepsilon$ if $i > j$ [26]. Let $\mathbb{N}_0$ be the set of non-negative integers.

Approximate String Matching (ASM): *ASM* problem considers the string matching problem with errors. It is an important problem in many branches of computer science, with several applications to text searching, computational biology, pattern recognition, signal processing etc. [9,23,26].

Let d: $\Sigma^* \times \Sigma^* \rightarrow \mathbb{N}_0$ be the *distance function*. The *distance* $d(x, y)$ between two strings $x = x[1..n] \in \Sigma^*$ and $y = y[1..m] \in \Sigma^*$ is the minimal cost of a sequence of operations that transform x into y (and ∞ if no such sequence exists).

The cost of a sequence of operations is the sum of the costs of the individual operations. In general, the set of possible operations are insertion, deletion, substitution or replacement and transposition [23]. Therefore under the distance measure, the *ASM problem* becomes minimizing the total cost to transform the pattern and its occurrence in a text to make them equal and find the text positions where this cost is low enough. Some of the most classical distance metrics are Levenshtein distance [22], Damerau distance [10] and Hamming distance [18]. Hamming Distance (HD), denoted as d_H, allows only replacements. It is restricted to equal length strings. In the literature, the search problem in many cases is called "string matching with mismatches" [23,26].

Since in this paper, the APAPSM problem is considered under HD, the following definitions are considered using HD applied to equal length strings. From now onwards we assume that $d = d_H$, for notational simplicity. Given an error threshold $k \in \mathbb{N}_0$, a pair of strings $u \in \Sigma_u^*$ and $v \in \Sigma_v^*$ where $m = |u| = |v|$, consider the following definitions. Without loss of generality, we presume both the alphabet sizes are equal when dealing with a bijection between the alphabets.

Parameterized String Matching (PSM): String $u = u[1..m]$ is said to be a *parameterized match* or *p-match* with v (denoted as $u \mathrel{\widehat{=}} v$) if there exists a bijection π: $\Sigma_u \to \Sigma_v$ such that $\pi(u) = \pi(u[1])\pi(u[2]) \ldots \pi(u[m]) = v$ [3].

Approximate Parameterized String Matching (Without Error Threshold): Given a bijection π: $\Sigma_u \to \Sigma_v$, the π-mismatch between u and v is the HD between the image of u under π and v, i.e., $d(\pi(u), v)$ [19]. We denote this by π-mismatch(u, v). Note that, there is an exponential number of possible bijections from Σ_u to Σ_v. Also, such π for which $d(\pi(u), v)$ is minimum, may not be unique.

The Approximate Parameterized String Matching (APSM) between u and v is to find a π such that over all bijections π-mismatch(u, v) is minimized. We denote this by $APSM(u, v)$. Formally, $APSM(u, v) = \{\pi \mid d(\pi(u), v)$ is minimum over all $\pi\}$. We define the *cost of* $APSM(u, v)$ as $cost(APSM(u, v)) = d(\pi(u), v)$ where $\pi \in APSM(u, v)$.

Parameterized String Matching (PSM) with k Mismatches: PSM with k mismatch seeks to find a bijection π: $\Sigma_u \to \Sigma_v$ such that the π-mismatch$(u, v) \leq k$. We then say that u parameterized matches v with k threshold. In literature, this problem is also known as *string comparison problem with threshold* k [19]. However, any π with π-mismatch$(u, v) \leq k$ will be satisfactory in this case (i.e., π-mismatch(u, v) need not be the minimum one over all π: $\Sigma_u \to \Sigma_v$).

Both the above problems were solved in $O(m^{1.5})$ time [19] by reducing them to maximum weight bipartite matching problem and using Kao et al.'s algorithm [20]. Let us define APSM problem with k error threshold as follows.

Approximate Parameterized String Matching with k Error Threshold: APSM with k error threshold, denoted as $APSM(u, v, k)$, seeks to find a π: $\Sigma_u \rightarrow \Sigma_v$ (over all bijections) such that $d(\pi(u), v)$ is minimum but not greater than k [19]. More formally, $APSM(u, v, k) = \{\pi \mid \pi \in APSM(u, v) \ \wedge \ d(\pi(u), v) \leq k\}$. We define *cost of* $APSM(u, v, k)$ as $cost(APSM(u, v, k)) = d(\pi(u), v)$, where $\pi \in APSM(u, v, k)$. In case, $APSM(u, v, k) = \emptyset$, then $cost(APSM(u, v, k)) = \infty$.

Example 1 in page 4 shows the difference between the above definitions.

3 All Pairs Approximate Parameterized String Matching

In this section, we investigate all pairs (best) approximate parameterized string matching (APAPSM) problem with k error threshold (with respect to Hamming distance error model) among the two sets P and T of equal length strings. The problem definition is the following along with the other required definitions[1].

Definition 1 (Pair Approximate Parameterized String Matching (PAPSM) with Error Threshold k). *Given a string $p \in \Sigma_P^m$ and $T = \{t_1, t_2, \ldots, t_{n_T}\} \subseteq \Sigma_T^m$, where $|\Sigma_P| = |\Sigma_T| = \sigma$ and $0 \leq k \leq m$. The PAPSM problem with k error threshold is to find j such that $APSM(p, t_j, k)$ gives π_j over all bijections and $d(\pi_j(p), t_j)$ is minimum over all j where $1 \leq j \leq n_T$.*

Denote this problem as $PAPSM(p, T, k)$. In more formal notation, $PAPSM(p, T, k) = \{j \mid \pi_j \in APSM(p, t_j, k) \ \wedge \ d(\pi_j(p), t_j) = \min_{1 \leq i \leq n_T}\{cost(APSM(p, t_i, k))\}\}$. In other words, the problem is to find $t_j \in T$ which is approximately parameterized closest to p with k error threshold. We call $d(\pi_j(p), t_j)$ as the *cost of* $PAPSM(p, T, k)$ and let us denote this by $cost(PAPSM(p, T, k))$. In case, $PAPSM(p, T, k) = \emptyset$, then $cost(PAPSM(p, T, k)) = \infty$.

Example 1. Given $p = abab \in \Sigma_P^4 = \{a, b\}^4$, $T = \{t_1 = cdcd, \ t_2 = dcdc, \ t_3 = ccdd, \ t_4 = cccd\} \subseteq \Sigma_T^4 = \{c, d\}^4$ and $k = 1$. Now,

$$
\begin{aligned}
&APSM(p, t_1, k) = \{\pi_1 = \{a \rightarrow c, b \rightarrow d\}\}, \quad d(\pi_1(p), t_1)) = 0;\\
&APSM(p, t_2, k) = \{\pi_2 = \{a \rightarrow d, b \rightarrow c\}\}, \quad d(\pi_2(p), t_2)) = 0;\\
&APSM(p, t_3, k) = \emptyset;\\
&APSM(p, t_4, k) = \{\pi_4 = \{a \rightarrow c, b \rightarrow d\}\}, \quad d(\pi_4(p), t_4)) = 1.
\end{aligned}
$$

Observe that, $\pi_3 = \{a \rightarrow c, \ b \rightarrow d\} \in APSM(p, t_3)$ but $d(\pi_3(p), t_3)) = 2 > k$. So, $APSM(p, t_3, 1) = \emptyset$. Hence, $PAPSM(p, T, 1) = \{1, 2\}$. Also note that, if $k = 3$, then for $\pi_4' = \{a \rightarrow d, b \rightarrow c\}$, π_4'-mismatch$(p, t_4) \leq k$. Hence just finding π_4' is also satisfactory to say that p is parameterized matched with t_4 under $k = 3$ error threshold; whereas $APSM(p, t_4, 3) = \{\pi_4\}$ and $\pi_4' \notin APSM(p, t_4, 3)$. □

[1] These definitions can also be extended with respect to other error models.

Note that it is sufficient to report a string from T which is closest to p under a given error threshold k. Also, it is possible to enumerate all $t_i \in T$ which are closest to p. Observe that, if $PAPSM(p, T, k) = \{i, j\}$ corresponding to the strings t_i and t_j, then $cost(PAPSM(p, T, k)) \leq k$ and more importantly, $cost(PAPSM(p, T, k)) = d(\pi_i(p), t_i) = d(\pi_j(p), t_j)$.

Theorem 1. *Given $p \in \Sigma_P^m$ and $T = \{t_1, t_2\} \subseteq \Sigma_T^m$. If p is an approximate parameterized matched with t_1 and $t_1 \widehat{=} t_2$, then p is also approximate parameterized matched with t_2 and its cost equal to $cost(APSM(p, t_1))$.*

Proof. The proof consists of two phases. Since p is approximate parameterized matched with t_1 (without any error threshold), then say $\pi_1 \in APSM(p, t_1)$. As a consequence, $cost(APSM(p, t_1)) = d(\pi_1(p), t_1)$ and moreover it is minimum over all bijections from Σ_P to Σ_T. Also, since $t_1 \widehat{=} t_2$, there exist a bijection, say π: $\Sigma_T \rightarrow \Sigma_T$ such that $\pi(t_1) = t_2$ and so $cost(APSM(t_1, t_2)) = d(\pi(t_1), t_2) = 0$.

Let $\pi_2 = \pi \circ \pi_1$: $\Sigma_P \rightarrow \Sigma_T$ and is defined as $\pi \circ \pi_1(u) = \pi(\pi_1(u))$ where $u \in \Sigma_P^m$. It can be easily proved by contradiction that $d(\pi_2(p), t_2)$ is minimum over all bijections. So we skip it.

Now, $cost(APSM(p, t_2)) = d(\pi_2(p), t_2) = d(\pi(\pi_1(p)), t_2) = d(\pi(\pi_1(p)), \pi(t_1))$ $= d(\pi_1(p), t_1) = cost(APSM(p, t_1))$. Therefore, $\pi_2 = \pi \circ \pi_1 \in APSM(p, t_2)$ and its cost equal to $cost(APSM(p, t_1))$ unit. □

The above theorem is extended for APSM problem with k error threshold.

Theorem 2. *Given $p \in \Sigma_P^m$ and $T = \{t_1, t_2\} \subseteq \Sigma_T^m$ and $0 \leq k \leq m$. If p is an approximate parameterized matched with t_1 under the k error threshold and $t_1 \widehat{=} t_2$, then p is also approximate parameterized matched with t_2 under the k error threshold and with the cost equal to $cost(APSM(p, t_1, k))$.*

Definition 2 (All Pairs Approximate Parameterized String Matching (APAPSM) with k Threshold). *Let $P = \{p_1, p_2, \ldots, p_{n_P}\} \subseteq \Sigma_P^m$ and $T = \{t_1, t_2, \ldots, t_{n_T}\} \subseteq \Sigma_T^m$. The problem is to find a mapping η: $[1, n_P] \rightarrow [1, n_T]$ such that sum of the $cost(APSM(p_i, t_{\eta(i)}, k))$ over all i $(1 \leq i \leq n_P)$ is minimum.*

Let us denote this problem as $APAPSM(P, T, k)$. The problem is to search: for each $p_i \in P$ $(1 \leq i \leq n_P)$, a $t_j \in T$ $(1 \leq j \leq n_T)$ which is approximately parameterized closest to p_i under k error threshold. More formally, $APAPSM(P, T, k) = \{(PAPSM(p_1, T, k), PAPSM(p_2, T, k), \ldots, PAPSM(p_{n_P}, T, k))\}$.

Theorem 3. *The above problem can be solved in $O(n_P\, n_T\, m^{1.5})$ time.*

Proof. It is direct from the solution of APSM problem proposed by Hazay et al. [19] by considering all possible pairs between P and T.

Definition 3 ($\gamma(k)$ -match of strings). *Let $k \in \mathbb{N}_0$. For two given strings $u = u[1..m]$, $v = v[1..m] \in \Sigma^*$ and the alphabet set $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$ where each $a_i \in \mathbb{N}_0$, u is said to be $\gamma(k)$-matched with v if and only if $\sum_{i=1}^{m} |u_i - v_i| \leq k$.*

The term $\gamma(k) - match$ is a suitably renamed version of the terminology $\gamma - approximate$ which was prescribed in [6] and defined on strings. Similar as above, we define $\gamma(k)$-match on two equal cardinality vectors of numbers.

Definition 4 (γ-distance, $\gamma(k)$-match of vectors). *Given two vectors $u = (u_1, u_2, \ldots, u_m)$, $v = (v_1, v_2, \ldots, v_m)$ where $u_i, v_j \in \mathbb{N}_0, 1 \leq i, j \leq m$ and $l, k \in \mathbb{N}_0$. γ-distance between u and v is l (denoted as $\gamma(u, v) = l$) if and only if $l = \sum_{i=1}^{m} |u_i - v_i|$. We say that u, $\gamma(k)$-matches with v, if and only if $\gamma(u, v) = \sum_{i=1}^{m} |u_i - v_i| \leq k$.*

The notion of Parikh mapping or vector was introduced by R.J. Parikh in [24]. It provides numerical properties of a string in terms of a vector by counting the number of occurrences of the symbols in the string. Parikh vector of a string w is denoted as $\psi(w)$.

Definition 5 (Parikh Vector (PV)). *Let $\Sigma = \{a_1, a_2, \ldots, a_\sigma\}$. Given $w \in \Sigma^*$, $\psi(w) = (f(a_1, w), f(a_2, w), \ldots, f(a_\sigma, w))$ where $f(a_i, w)$ gives the frequency of the symbol $a_i \in \Sigma$ $(1 \leq i \leq \sigma)$ in the string w.*

For example, if $\Sigma = \{c, d\}$ then $\psi(cddcc) = (3, 2)$. However, much information is lost in the transition from a string to its PV. Note that Parikh mapping is not injective as many strings over an alphabet may have the same PV and so the information of a string is reduced while changing the string to a PV. For example, the strings *ccccdddd* and *dcdcdcd* have the same Parikh vector $(3, 4)$.

Definition 6 (Normalized Parikh Vector (NPV)). *NPV of a string $w \in \Sigma^*$ is $\widehat{\psi}(w) = (g_1, g_2, \ldots, g_\sigma)$ such that $\forall i,\ 1 \leq i < \sigma,\ g_i \geq g_{i+1}$ and there exists a bijective mapping ρ: $\{1..\sigma\} \rightarrow \{1..\sigma\}$ such that $g_i = f(a_{\rho(i)}, w)$.*

In other words, we sort the elements of $\psi(w)$ in non-increasing order to get the $\widehat{\psi}(w)$ of string w. For example, $\psi(dcdcdcd) = (3, 4)$ and $\widehat{\psi}(dcdcdcd) = (4, 3)$.

Theorem 4. *Given a pair of equal length strings $u \in \Sigma_P^*$ and $v \in \Sigma_T^*$, if $u \mathrel{\widehat{=}} v$ then $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = 0$.*

Proof. Since $u \mathrel{\widehat{=}} v$, then by definition there exists a bijection π: $\Sigma_P \rightarrow \Sigma_T$ such that $\pi(u) = v$, i.e. $\pi(u)$ is obtained by renaming each character of u using π. Though symbols of Σ_P are renamed by π, the frequency of each symbol $a \in \Sigma_P$ in u will be same as the frequency of $\pi(a) \in \Sigma_T$ in $v = \pi(u)$. As a consequence, $\widehat{\psi}(v) = \widehat{\psi}(u)$, even though there may be the case $\psi(u) \neq \psi(v)$. □

However, the converse is not always true. To show that, we shall give the following example.

Example 2. Given $p = ababa, \in \Sigma_P^* = \{a, b\}^*$ and $T = \{t_1 = cdcdd,\ t_2 = dcdcd\} \subseteq \Sigma_T^* = \{c, d\}^*$. Now,

$$\psi(p) = (3, 2) \text{ and } \widehat{\psi}(p) = (3, 2);$$
$$\psi(t_1) = (2, 3) \text{ and } \widehat{\psi}(t_1) = (3, 2);$$
$$\psi(t_2) = (2, 3) \text{ and } \widehat{\psi}(t_2) = (3, 2).$$

As mentioned in Theorem 4, $p \mathrel{\widehat{=}} t_2$ and so $\gamma(\widehat{\psi}(p), \widehat{\psi}(t_2)) = 0$, even though $\psi(p) \neq \psi(t_2)$. Conversely, $\widehat{\psi}(t_1) = \widehat{\psi}(p) = \widehat{\psi}(t_2) = (3, 2)$, but $p \not\mathrel{\widehat{=}} t_1$ and $p \mathrel{\widehat{=}} t_2$. Hence, in case $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = 0$, it is required to check if $u \mathrel{\widehat{=}} v$ or not. □

In general, a *filter* is a device or subroutine that processes the feasible inputs and tries to remove some undesirable component. We design an interesting filtering technique by using Parikh vector in order to preprocess the given strings of P and T and to reduce the number of pair comparisons for solving approximate parameterized string matching between the pair of strings under k error threshold. We name the filter which is mentioned in Theorem 7 as *PV-filter* and the process of filtering the input data by PV-filter as *PV-filtering*.

The following theorems are useful in minimizing the number of pairs comparisons for APAPSM problem to improve the solution from the practical aspect. Theorem 5 is applicable for ASM problem. It is extended in Theorems 6 and 7 in the context of APSM problem without and with k error threshold, respectively.

Theorem 5. *Let $u, v \in \Sigma^*$ be a pair of equal length strings and $k = d(u, v)$, is the Hamming distance. Then $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) \leq 2k$ and $\gamma(\psi(u), \psi(v)) \leq 2k$.*

Proof. We prove it by the principle of mathematical induction on k.

Base case: For $u = v$, $k = d(u, v) = 0$ and $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = 0$.
Hypothesis: Assume that for any k with $0 \leq k = d(u, v) \leq i$, $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) \leq 2k$.
Inductive step: Let, after introducing one more error by replacement (symbol $a \in \Sigma$ is replaced by $b \in \Sigma$ in any position of u) operation in u we get u' such that $d(u', u) = 1$ and $k = d(u', v) = i + 1$. However, while changing u to u' with $d(u', u) = 1$, there may be only other case that $k = d(u', v) = i - 1$ for which also the inequality is true (by the induction hypothesis). So we have to argue for the former case: $k = i + 1$. While introducing one error by replacement, $\gamma(\widehat{\psi}(u'), \widehat{\psi}(u))$ will be increased by at most 2 as the frequency of symbol a is decreased by one and the frequency of b is increased by one. Hence, $\gamma(\widehat{\psi}(u'), \widehat{\psi}(v)) \leq 2i + 2 = 2(i + 1)$ while $k = d(u', v) = i + 1$.

Hence the proof of the first inequality, by the principle of mathematical induction.

For the other one also, the proof justification is similar. □

Theorem 6. *Given a pair of strings $u \in \Sigma_P^m$, $v \in \Sigma_T^m$, let $k = cost(APSM(u, v))$. Then $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) \leq 2k$.*

Proof. Let $\pi \in APSM(u, v)$. Therefore by definition, $k = cost(APSM(u, v)) = d(\pi(u), v)$ is minimum over all bijections. Let $\pi(u) = u' \in \Sigma_T^m$. Since $u \mathrel{\widehat{=}} u'$ under π, $\widehat{\psi}(u) = \widehat{\psi}(u')$, by Theorem 4. Hence $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = \gamma(\widehat{\psi}(u'), \widehat{\psi}(v))$. By using Theorem 5, we have $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = \gamma(\widehat{\psi}(u'), \widehat{\psi}(v)) \leq 2k$. □

Theorem 7. *Given $u \in \Sigma_P^m$ and $v \in \Sigma_T^m$. Let $\widehat{k} = cost(APSM(u, v, k))$. Then $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) \leq 2\widehat{k}$ (which we call as PV-filter).*

Proof. The proof is very similar as Theorem 6. Let $\pi \in APSM(u, v, k)$. Accordingly, there exists a bijection π: $\Sigma_P \to \Sigma_T$ such that $\widehat{k} = cost(APSM(u, v, k)) = d(\pi(u), v)$ is minimum but not greater than k. Let $u' = \pi(u) \in \Sigma_T^m$. With similar argument as above, we have $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = \gamma(\widehat{\psi}(u'), \widehat{\psi}(v)) \leq 2\widehat{k}$. □

We use this PV-filter as a subroutine during the design of a simple algorithm to solve the APAPSM problem with error threshold k between two sets P and T of equal length strings. In worst-case (i.e. none of the pairs are filtered out by PV-filter), it takes $O(n_P\, n_T\, m)$.

Computing APAPSM Under Error Threshold: Let $P = \{p_1,\ p_2, \ldots, p_{n_P}\} \subseteq \Sigma_P^m$ and $T = \{t_1,\ t_2, \ldots, t_{n_T}\} \subseteq \Sigma_T^m$ be two sets of strings where $|\Sigma_P| = |\Sigma_T| = \sigma$. In Algorithm 1, we compute APAPSM problem with error threshold $k \in \mathbb{N}_0$ among two sets P and T of equal length strings. In Step 3, clustering is precisely recommended, in case in advance it is known that there are many exact and parameterized repetition of strings in P and T. To create the equivalence classes in P and T separately, with respect to parameterization, clustering is done based on the converse of Theorem 4, i.e., in case for any two strings $u, v \in P$ (and T, respectively) if $\gamma(\widehat{\psi}(u), \widehat{\psi}(v)) = 0$, then and only then check for $u \mathrel{\widehat{=}} v$. If $u \mathrel{\widehat{=}} v$ holds, them put u and v into the same cluster.

Algorithm 1. Compute $APAPSM(P, T, k)$ after using the PV-filter

Input: The sets P, T of equal length strings and an error threshold k.
Output: $APAPSM(P, T, k)$ with respect to Hamming distance error model.

APAPSM(P, T, k)
1: **for** $i \leftarrow 1 : n_P$ **do** compute NPV of p_i.
2: **for** $i \leftarrow 1 : n_T$ **do** compute NPV of t_i.
3: **do** parameterized clustering of P and T, i.e., for any $(p_1, p_2) \in P \times P$ or $T \times T$ of a cluster, $p_1 \mathrel{\widehat{=}} p_2$. To speed up the clustering, **if** $\gamma(\widehat{\psi}(p_1), \widehat{\psi}(p_2)) = 0$ **then** only check for $p_1 \mathrel{\widehat{=}} p_2$, or otherwise, $p_1 \mathrel{\widehat{\neq}} p_2$ (Negation of Theorem 4).
4: **for** each parameterized cluster of P, pick a representative, say p_i
  for each parameterized cluster of T, pick a representative, say t_j
    if $\gamma(\widehat{\psi}(p_i), \widehat{\psi}(t_j)) \leq 2k$ (which is the *PV-filtering*)
      then compute $APSM(p_i, t_j, k)$.
    end if
  end for
end for

Complexity Analysis: Steps 1–3 of Algorithm 1 are the preprocessing steps for computing $APAPSM(P, T, k)$; Steps 1–2 takes $O(m(n_P + n_T))$ and Step 3 takes $O(m(n_P^2 + n_T^2))$ time, assuming a constant size alphabets. But as mentioned earlier, clustering is optional, it might be skipped depending on the circumstances.

In Step 4, for any pair $(p_i, t_j) \in P \times T$, computation of $APSM(p_i, t_j, k)$ can be done by reducing the problem to maximum weight bipartite matching (MWBM) problem [19]. Let $G = (V, E, W)$ be an undirected, weighted (non-negative integer weight) bipartite graph where V, E and W are the vertex set, edge set and total weight of G, respectively. MWBM problem can be solved in $O(\sqrt{|V|}W')$ time, where $|E| \leq W' \leq W$ [12]. It is a fine-tuned version of the existing decomposition solution [20]. Using the fine-tuned decomposition solution for MWBM, $APSM(p_i, t_j, k)$ can be solved in $O(m\sqrt{\sigma})$ where $W' = O(W) = O(m)$ and $V = O(\sigma)$ [12,13]. In the worst-case scenario: each of the clusters will have just a single string either from P or T and PV-filter in Step 4 does not filter out any pair $(p_i, t_j) \in P \times T$. Therefore worst-case running time of the Algorithm 1 is $O(n_P\, n_T\, m\sqrt{\sigma})$, which is $O(n_P\, n_T\, m)$, if we assume a constant alphabet.

4 Experimental Results

To test the efficiency of the PV-filter, we performed several experimental studies, but only a few are reported in this section because of page limitation. Algorithm 1 which solves $APAPSM(P, T, k)$, is implemented in *MATLAB Version 7.8.0.347 (R2009a)*. All the experiments are conducted on a PC Laptop with an *Intel® Core™ 2 Duo (T6570 @ 2.10 GHz) Processor, 3.00 GB RAM* and *500 GB Hard Disk*, running the *Microsoft Windows 7 Ultimate (32-bit Operating System)*.

Data Description: We generate the input data sets P and T by using the pre-defined `randi` function. It helps to generate uniformly distributed pseudorandom integers. The function `randi(imax,m,n)` returns an `m`-by-`n` matrix containing pseudorandom integer values drawn from the discrete uniform distribution on `1:imax`.

Efficiency of PV-Filter: The experimental results show that the PV-filter is efficient, essentially for small error threshold k, to avoid unwanted pairs (u, v) comparison for $APSM(u, v, k)$, where $u \in P$ and $v \in T$. According to the random experiment, if the error threshold $k \leq \frac{m}{3}$, then almost more than one-third of the total pairs comparison can be skipped. Moreover, very smaller threshold gives much better filtering. Please see the experiments.

Experiment 1. *Consider, alphabet sets* $\Sigma_P = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}\}$, $\Sigma_T = \{\mathtt{a}', \mathtt{b}', \mathtt{c}', \mathtt{d}', \mathtt{e}', \mathtt{f}', \mathtt{g}', \mathtt{h}', \mathtt{i}', \mathtt{j}'\}$; $P \in \Sigma_P^*$, $T \in \Sigma_T^*$; *cardinality of each of the sets* P *and* T *is* $|P| = |T| = 100$; *and* $|p_i| = |t_j| = 6$ *for* $1 \leq i, j \leq |P| = |T|$.

According to the data set generated in Experiment 1, a total of 10,000 (u, v) pairs of comparisons for $APSM(u, v, k)$, where $u \in P$ and $v \in T$, are required without PV-filtering. Figure 1 shows the efficiency graph of the filter on the input data set. Each blue "$*$" point in the graph indicates the number of elimination of pairs comparison for a given error threshold, after using the PV-filter.

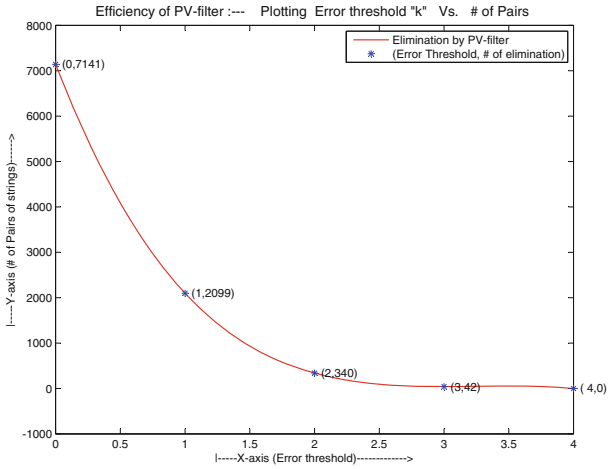

Fig. 1. Elimination graph of pairs of strings after using PV-filter for the input data set with $|\Sigma_P| = |\Sigma_T| = 10$; $|P| = |T| = 100$; $|p_i| = |t_j| = 6$ for $1 \leq i, j \leq |P| = |T|$, as mentioned in Experiment 1.

Table 1. PV-filtering for the data set in Experiment 1.

Number of pairs ↓	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
Before using PV-filter	10,000	10,000	10,000	10,000	10,000
Eliminated by PV-filter	7,141	2,099	340	42	0
Passed through PV-filter	2,859	7,901	9,660	9,958	10,000
Whose APSM cost $\leq k$	570	3,986	8,699	9,869	10,000

For example, each point (i, j) in Fig. 1 represents that for $k = i$ error threshold, j number of (u, v) pairs of strings have skipped the comparison for $APSM(u, v, i)$.

Table 1 gives more light to the Experiment 1. The second row represents that for a given k, a total number of (u, v) pairs are to be checked for $APSM(u, v, k)$, initially before using PV-filter; the third row says, for respective k the number of pairs of strings are eliminated by PV-filter; simultaneously, the fourth row describes that how many string pairs are passed by the filter; and finally, the last row mentions, for how many (u, v) pairs, actually $cost(APSM(u, v)) \leq k$ among the passed pairs.

Experiment 2. *Consider the alphabet sets* $\Sigma_P = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{x}, \mathtt{y}, \mathtt{z}\}$, $\Sigma_T = \{\mathtt{a}', \mathtt{b}', \mathtt{c}', \ldots, \mathtt{x}', \mathtt{y}', \mathtt{z}'\}$ *with* $|\Sigma_P| = |\Sigma_T| = 26$; $P \in \Sigma_P^*$, $T \in \Sigma_T^*$; *cardinality of the sets* P *and* T *is* $|P| = |T| = 100$; *and* $|p_i| = |t_j| = 2000$ *for* $1 \leq i, j \leq |P| = |T|$. *Figure 2 gives the elimination graph. The corresponding table is skipped due to space limitation.*

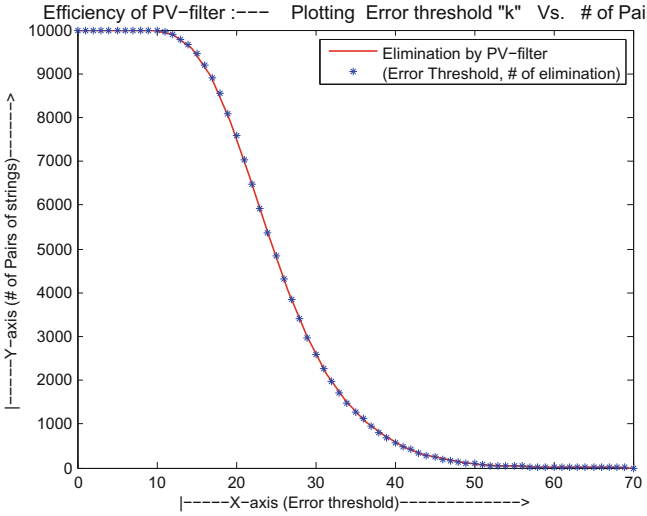

Fig. 2. Elimination graph of pairs of strings after using PV-filter for the input data set with $|\Sigma_P| = |\Sigma_T| = 26$; $|P| = |T| = 100$; $|p_i| = |t_j| = 2000$ for $1 \leq i, j \leq |P| = |T|$, as mentioned in Experiment 2.

5 Conclusions

In this paper, we have explored all pairs approximate parameterized string matching problem with k Hamming distance error threshold between two sets of equal length stings. We have presented a solution with worst-case complexity $O(n_P\, n_T\, m)$, assuming constant alphabet size. In order to minimize number of paired comparisons for solving APSM between pair of strings with error threshold, we have proposed a PV-filtering technique by using Parikh vector. Although the filter does not improve the worst-case asymptotic bound, but the using it as a subroutine, we can avoid some of the unwanted paired comparisons for APSM. Experimental results show that the PV-filter is efficient for small error threshold.

Acknowledgement. The author is grateful to Dr. Jan Holub for his helpful comments and suggestions.

References

1. Apostolico, A., Erdős, P.L., Jüttner, A.: Parameterized searching with mismatches for run-length encoded strings. Theor. Comput. Sci. **454**, 23–29 (2012). Formal and Natural Computing Honoring the 80th Birthday of Andrzej Ehrenfeucht
2. Apostolico, A., Erdős, P.L., Lewenstein, M.: Parameterized matching with mismatches. J. Discret. Algorithms **5**(1), 135–140 (2007)

3. Baker, B.S.: A theory of parameterized pattern matching: algorithms and applications. In: Symposium on Theory of Computing, pp. 71–80. ACM (1993)
4. Baker, B.S.: Parameterized pattern matching: algorithms and applications. J. Comput. Syst. Sci. **52**(1), 28–42 (1996)
5. Baker, B.S.: Parameterized duplication in strings: algorithms and an application to software maintenance. SIAM J. Comput. **26**(5), 1343–1362 (1997)
6. Cambouropoulos, E., Crochemore, M., Iliopoulos, C.S., Mouchard, L., Pinzon, Y.J.: Algorithms for computing approximate repetitions in musical sequences. Int. J. Comput. Math. **79**(11), 1135–1148 (2002)
7. Charras, C., Lecroq, T.: Handbook of Exact String Matching Algorithms. King's College Publications, London (2004)
8. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, New York (2007)
9. Crochemore, M., Rytter, W.: Jewels of Stringology: Text Algorithms. World Scientific Press, Singapore (2002)
10. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM **7**(3), 171–176 (1964)
11. Das, S., Holub, J., Kapoor, K.: All pairs approximate parameterized string matching. Technical report FIT-2014-01, Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Thákurova 2700/9, 160 00 Praha 6, Czech Republic, March 2014
12. Das, S., Kapoor, K.: Fine-tuning decomposition theorem for maximum weight bipartite matching. In: Gopal, T.V., Agrawal, M., Li, A., Cooper, S.B. (eds.) TAMC 2014. LNCS, vol. 8402, pp. 312–322. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06089-7_22
13. Das, S., Kapoor, K.: Weighted approximate parameterized string matching. AKCE Int. J. Graphs Comb. **14**(1), 1–12 (2017)
14. Faro, S., Lecroq, T.: The exact online string matching problem: a review of the most recent results. ACM Comput. Surv. **45**(2), 13:1–13:42 (2013)
15. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3), 596–615 (1987)
16. Gabow, H.N.: Scaling algorithms for network problems. J. Comput. Syst. Sci. **31**(2), 148–168 (1985)
17. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM J. Comput. **18**(5), 1013–1036 (1989)
18. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
19. Hazay, C., Lewenstein, M., Sokol, D.: Approximate parameterized matching. ACM Trans. Algorithms **3**(3) (2007). https://doi.org/10.1145/1273340.1273345
20. Kao, M.Y., Lam, T.W., Sung, W.K., Ting, H.F.: A decomposition theorem for maximum weight bipartite matchings. SIAM J. Comput. **31**(1), 18–26 (2001)
21. Lee, I., Mendivelso, J., Pinzón, Y.J.: $\delta\gamma$ – parameterized matching. In: Amir, A., Turpin, A., Moffat, A. (eds.) SPIRE 2008. LNCS, vol. 5280, pp. 236–248. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89097-3_23
22. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
23. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**, 31–88 (2001)
24. Parikh, R.J.: On context-free languages. J. ACM **13**(4), 570–581 (1966)

25. Prasad, R., Agarwal, S.: Study of bit-parallel approximate parameterized string matching algorithms. In: Ranka, S., Aluru, S., Buyya, R., Chung, Y.-C., Dua, S., Grama, A., Gupta, S.K.S., Kumar, R., Phoha, V.V. (eds.) IC3 2009. CCIS, vol. 40, pp. 26–36. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03547-0_4
26. Smyth, B.: Computing Patterns in Strings. Pearson Addison-Wesley, New York (2003)