5 QUASAR - q-gram based database searching

This exposition has been developed by Knut Reinert and Clemens Gröpl. It is based on the following sources, which are all recommended reading:

1. Burkhardt et al. (1999) *q-gram Based Database Searching Using a Suffix Array (QUASAR)*, Proc. RECOMB 99.
2. Burkhardt and Kärkkäinen (2001) *Better Filtering with Gapped q-grams*, Proc. CPM 01.

The tool *QUASAR* aims at aligning a query $S = s_1, \ldots, s_m$ in a text, also called database $D = d_1, \ldots, d_n$. It can be seen as an efficient filter that uses exact matches. In contrast to online filtering algorithms, QUASAR uses a suffix array as indexing structure for the database.

5.1 Quasar

QUASAR, or "Q-gram Alignment based on Suffix ARrays", is a filtering approach. QUASAR finds all *local* approximate matches of a *query sequence* S in a *database* $D = \{d, \ldots\}$. The verification is performed by other means.

Definition. A sequence d is *locally similar* to S, if there exists at least one pair $(S_{i,i+w-1}, d')$ of substrings such that:

1. $S_{i,i+w-1}$ is a substring of length w and d' is a substring of D, and
2. the substrings d' and $S_{i,i+w-1}$ have edit distance at most k.

We call this the *approximate matching problem with k differences and window length w.*

For simplicity, we assume that the database consists of only one sequence, i. e. $D = \{d\}$.

5.2 The q-gram lemma

A short subsequence of length q is called a *q-gram*. In the following we start by considering the first w letters of S. The algorithm uses the following lemma:

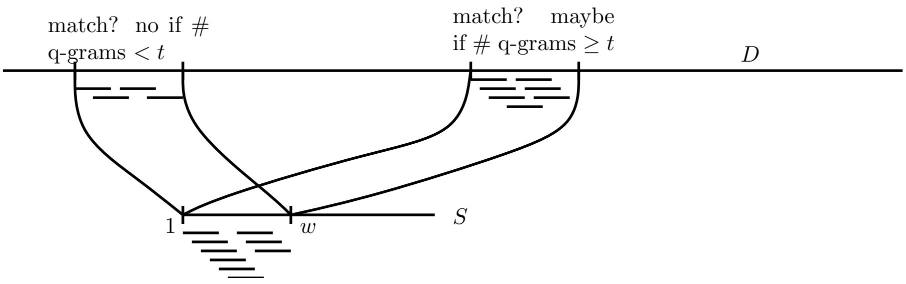
Lemma 1. *Let P and S be strings of length w with at most k differences. Then P and S share at least* $w + 1 - (k + 1)q$ *common q-grams.*

In our case, this means:

Lemma 2. *Let an occurrence of* $S_{1,w}$ *with at most k differences end at position j in D. Then at least* $w + 1 - (k + 1)q$ *of the q-grams in* $S_{1,w}$ *occur in the substring* $D_{j-w+1,j}$.

Proof: Exercise. . . .

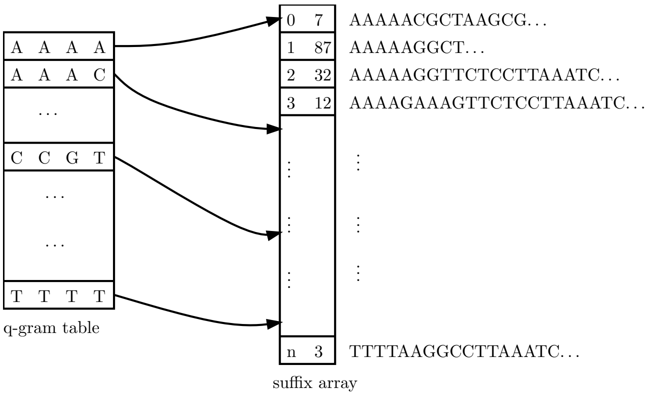
That means that as a necessary condition for an approximate match, at least $t = w + 1 - (k+1)q$ of the q-grams contained in $S_{1,w}$ occur in a substring of D with length w. For example the strings `ACAGCTTA` and `ACACCTTA` have $8 + 1 - (1 + 1)3 = 3$ common 3-grams, namely `ACA`, `CTT` and `TTA`.



5.3 q-gram index

The algorithm builds in a first step an indexing structure as follows:

1. Build a suffix array A over D.
2. Given q, compute for all possible $|\Sigma|^q$ q-grams the start position of the hitlist. This allows to lookup a q-gram in constant time.
3. If another q is specified, A is used to recompute the above table.

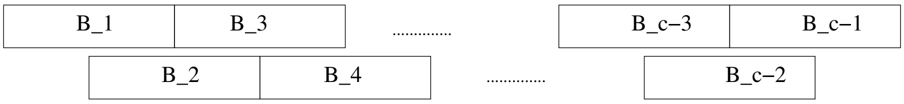


5.4 Counting q-grams

Now we have to find all approximate matches between $S_{1,w}$ and D, that means we have to find all substrings in D that share at least t q-grams with $S_{1,w}$. The algorithm proceeds in the following basic steps on which we will elaborate:

1. Define two arrays of *non-overlapping* blocks of size $b \geq 2w$. The first array is shifted by $b/2$ against the other.
2. Process all q-grams in $S_{1,w}$ and increment the counters of the corresponding blocks.
3. All blocks containing approximate matches will have a counter of at least t. (The reverse is not true).
4. Shift the search window by one. Now we consider $S_{2,w+1}$.

5.5 Blocking



Since we want to count the q-grams that are in common between the query and the database, we use counters. Ideally we would use a counter of size w for each substring of this size. Since this uses too much memory, we build larger, non-overlapping blocks. While this decreases the memory usage, it also decreases the specificity.

Since the blocks are not overlapping we might miss q-grams that cross the block boundary. As a remedy, we use a second, shifted array of blocks.

5.6 Window Shifting

We started the search for approximate matches of window length w with the first w-mer in S, namely $S_{1,w}$. In order to determine the approximate matches for the next window $S_{2,w+1}$, we only have to discard the old q-gram $S_{1,q}$ and consider the new q-gram $S_{w-q+2,w+1}$.

To do that we decrement the counters of all blocks that contain $S_{1,q}$ that have not reached the threshold t. However, if the counter has already reached t it stays at this value to indicate a match for the extension phase.

For the new block we use the precomputed index and the suffix array to find the occurrences of the new q-gram and increment the corresponding block counters (at most two).

5.7 Alignment

After having computed the list of blocks, QUASAR uses BLAST to actually search the blocks. Here are some results from the inital implementation. QUASAR was run with $w = 50$, $q = 11$, and t such that windows with at most 6% differences are found. Reasonable values for the block size are 512 to 4096.

DB size	query	id. res.	filtr. ratio	QUASAR	BLAST
73.5 Mb	368	91.4%	0.24%	0.123 s	3.27 s
280 Mb	393	97.1%	0.17%	0.38 s	13.27 s

"A database in BLAST format is built in main memory which is then passed to the BLAST search engine. The construction of this database requires a significant amount of time and introduces unnecessary overhead."

5.8 Gapped q-grams

In order to achieve a high filtration rate, we would like to choose q as large as possible, since the number of hits decreases exponentially in q. On the other hand, the threshold $t = w - q - qk + 1$ also decreases with increasing q thereby reducing the filtering efficiency. The question is whether we could increase the length of the q-grams somehow, such that the threshold t stays high.

This can indeed be achieved by using *gapped q-grams*. For example the 3-grams with the *shape* `##.#` in the string `ACAGCT` are `AC.G`, `CA.C`, and `AG.T`:

```
ACAGCT
AC G
 CA C
  AG T
```

Next we define the concept formally.

Definition 3.

- A *shape* Q is a set of non-negative integers containing 0.
- The *size* of Q, denoted by $|Q|$, is the cardinality of the set.
- The *span* of Q is $s(Q) = \max Q + 1$.
- A shape of size q and span s is called (q,s)*-shape*.
- For any integer i and shape Q, the *positioned shape* Q_i is the set $\{i + j \mid j \in Q\}$.
- Let $Q_i = \{i_1, i_2, \ldots, i_q\}$, where $i = i_1 < i_2 < i_3 < \cdots < i_q$, and let $S = s_1 s_2 \ldots s_m$ be a string. For $1 \leq i \leq m - s(Q) + 1$, the *Q-gram at position i in S*, denoted by $S[Q_i]$, is the string $s_{i_1+1} s_{i_2+1} \ldots s_{i_q+1}$.
- Two strings P and S *have a common Q-gram at position i* if $P[Q_i] = S[Q_i]$.

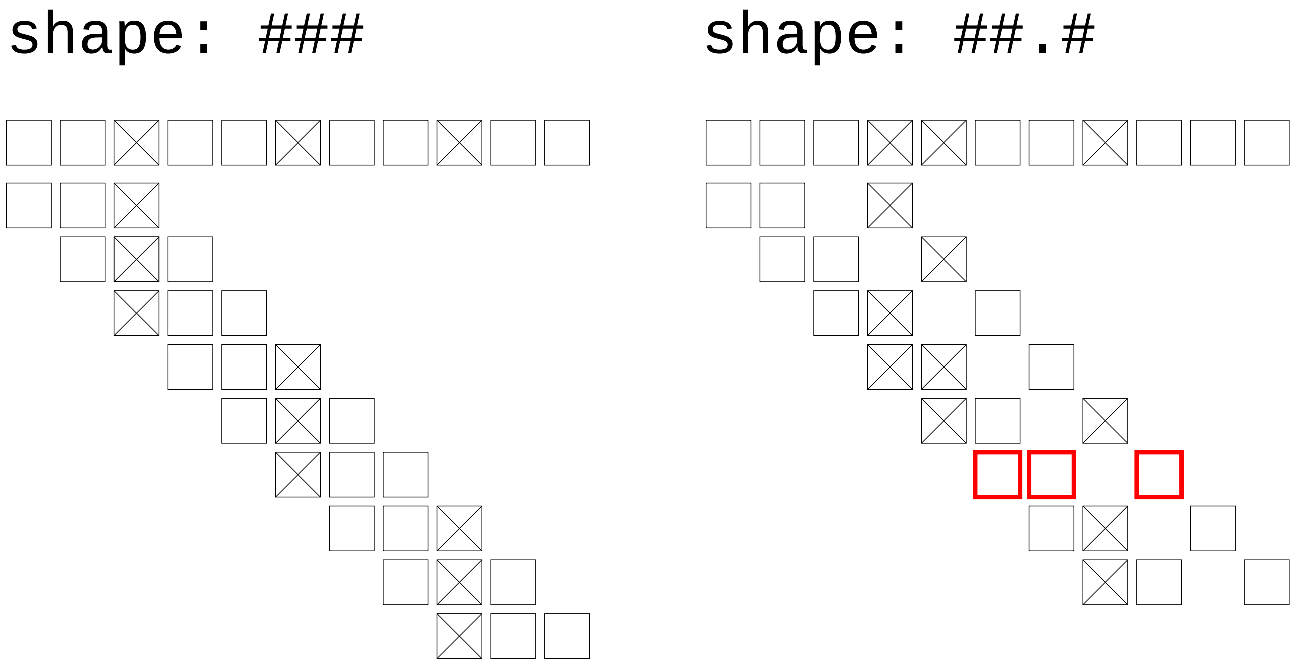
Example 4. Let $Q = \{0, 1, 3, 6\}$ be a shape. Using the graphical representation it is the shape `##.#..#`. Its size is $|Q| = 4$ and its span is $s(Q) = 7$. The string *ACGGATTAC* has three Q-grams: $S[Q_1] = s_1 s_2 s_4 s_7 = ACGT$, $S[Q_2] = CGAA$, and $S[Q_3] = GGTC$.

The q-gram lemma can be extended for gapped q-grams. A generalization gives

$$t = w - s(Q) - |Q|k + 1.$$

However it is not tight anymore (we will prove this).

Example 5. Let $w = 11$ and $k = 3$ and consider the 3-shapes `###` and `##.#`. The above threshold for the two shapes is $0 = 11 - 3 \cdot 4 + 1$ and $-1 = 11 - 4 - 3 \cdot 3 + 1$ respectively. Thus neither shape would be useful for filtering. However, the real threshold for `##.#` is 1. This can be checked by a full enumeration of all combinations of 3 mismatches.



Worst-case mismatch positions

5.9 New threshold

What is the (tight) threshold for arbitrary Q-shapes?

Let $P = p_1, \ldots, p_w$ and $S = s_1, \ldots, s_w$ be two strings of length w. Let $R(P, S)$ be the set of positions where S and P do not match. Then $|R(S, P)|$ is the Hamming distance of P and S.

To determine the common Q-grams of P and S only the mismatch set is needed: It holds that

$$P[Q_i] = S[Q_i] \quad \text{if and only if} \quad Q_i \cap R(P, S) = \emptyset.$$

Using this notation we can define the threshold of a shape Q for a pattern of length w and Hamming distance k as:

$$t(Q, w, k) := \min_{R \subseteq \{1,\ldots,w\}, |R|=k} \left|\{i \in \{1, \ldots, w - s(Q) + 1\} \mid Q_i \cap R = \emptyset\}\right|$$

From the above discussion we get the following tight form of the q-gram lemma for arbitrary shapes:

Lemma 6. *Let Q be a shape. For any two strings P and S of length w with Hamming distance k, the number of common Q-grams of P and S is at least $t(Q, w, k)$. Furthermore, there exist two strings P and S of length w and Hamming distance k, for which the number of common Q-grams is exactly $t(Q, w, k)$.*

It is easy to see that this bound is as least as tight as the lower bound we already introduced:

Lemma 7.

$$t(Q, w, k) \geq \max\{0, w - s(Q) - |Q|k + 1\}$$

Proof: Let R be the set minimizing the expression in the definition of $t(Q, w, k)$. For each $j \in R$ there are at most $|Q|$ integers i such that $j \in Q_i$. Therefore, at most $k|Q|$ of the positioned shapes Q_i, $i \in \{1, \ldots, w - s(Q) + 1\}$, intersect with R, and at least $w - s(Q) - k|Q| + 1$ do not intersect with R.

T The above lemma gives indeed the exact threshold for ungapped q-grams.

Lemma 8. *Let Q be a contiguous shape,i. e., $Q = \{0, \ldots, q - 1\}$. Then*

$$t(Q, w, k) = \max\{0, w - s(Q) - |Q|k + 1\} = \max\{0, w - q(k + 1) + 1\}.$$

Proof: The lower bound is shown by Lemma **??**. For the upper bound we choose $R = \{q, 2q, \ldots, kq\}$. Then Q_i intersects with R if and only if $i \in \{1, \ldots, kq\}$, and thus does not intersect with R if $i \in \{kq + 1, \ldots, w - q + 1\}$. Hence for this R we have only $w - q + 1 - kq - 1 + 1 = w - (k + 1)q + 1$ common q-grams.

The following table gives the exact thresholds for all shapes for $w = 50$ and $k = 5$. One can see that in many cases, especially for higher values of q, best gapped shapes have higher thresholds than contiguous shapes of the same or even smaller size.

$s \downarrow \ : \ q \rightarrow$	4	5	6	7	8	9	10
5	26	21	–	–	–	–	–
6	25	20	15	–	–	–	–
7	24	19	14	9	–	–	–
8	23	18	13	8	3	–	–
9	22	**18** > 17	**14** > 12	**9** > 7	**5** > 2	0	–
10	21	**18** > 16	**13** > 11	**10** > 6	**6** > 1	**3** > 0	0
11	20	**16** > 15	**13** > 10	**10** > 5	**7** > 0	**4** > 0	**2** > 0
12	19	**16** > 14	**12** > 9	**9** > 4	**7** > 0	**4** > 0	**2** > 0

It has to be noted that it does not suffice to put in gaps somewhere; the gaps have to be choosen carefully. For example in the above table ($w = 50$, $k = 5$, and $q = 12$) there are only two shapes with a positive threshold, namely `###.#..###.#..###.#` and `#.#.#...#.....#.#.#...#.....#.#.#...#` and their mirror images.

5.10 Minimum coverage

The filtering efficiency of a Q-gram clearly depends on the threshold $t(Q, w, k)$. However there is also another factor that influences it. This factor is called *minimum coverage*.

Before we define it formally lets have a look at an example.

Example 9. Let $w = 13$ and $k = 3$. Then both shapes `###` and `##.#` have a threshold of two. If two strings have four consecutive characters then they have two common 3-grams of shape `###`. In contrast, in order to have two common 3-grams of shape `##.#`, two strings need at least 5 matching characters.

This means, that the gapped 3-gram would have a lower count of common q-grams on strings that have only four consecutively matching characters although it has the same threshold.

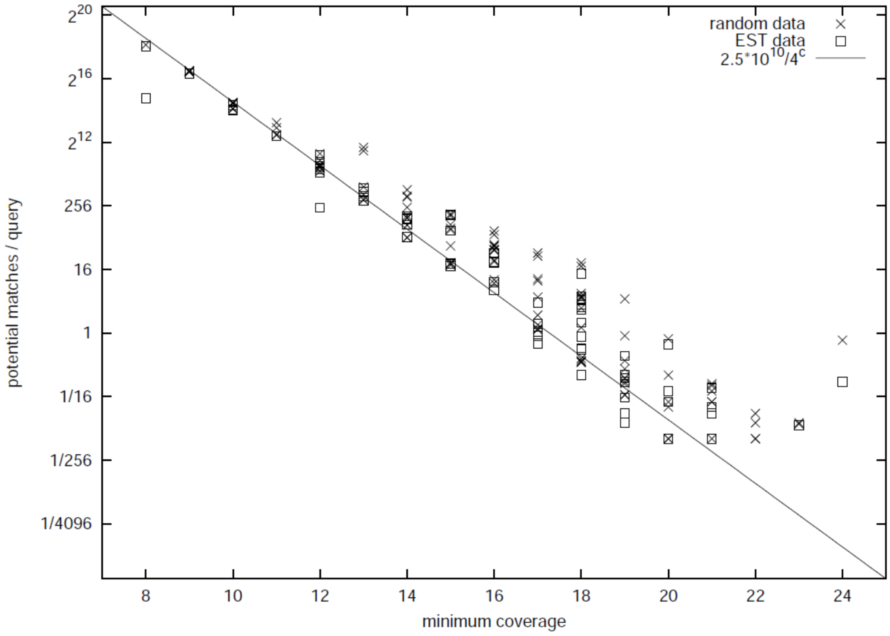
Definition 10. Let Q be a shape and t be a non-negative integer. The *minimum coverage* of Q for threshold t is:

$$c(Q, t) = \min_{C \subset \mathbb{N}, |C| = t} | \cup_{i \in C} Q_i | .$$

Hence the minimum coverage is the minimum number of characters that need to match between a pattern and a text substring for there to be t matching Q-grams.

Whenever possible, gapped Quasar chooses the highest minimum coverage, since it makes it more unlikely that a random string matches t Q-grams. This improves the filter efficency.

Computational experiments indicate that there is a strong correlation between the minimum coverage $c(Q, t(Q, m, k))$ and the filter efficiency.



Correlation between expected and actual number of potential matches.

The following table shows different shapes for $k = 5$. The column *best* shows the shape with the highest minimum coverage (ties are broken using the threshold). The column *median* shows the median shape ordered by minimum coverage. If one chooses a random shape, the chance is 50% to be better (or worse) than this one. The last column show the best *one-gapped* shape. (The details of the tie breaking used here can be read in the paper.)

q	best	median	1-gapped
6	##......#..#..#.#	#.###.....#.#	#####....#
7	#.##......##..#.#	##..#..#..##	#####...##
8	##..#.#...............##..#.#	#.#..####...#.#	######...##
9	###..#..#.#...#.##	######..#.#.#	#######..##
10	###..#..#.#..###.#	##.##..#.#.###.#	#######

5.11 Index structure

It is not necessary to use a suffix array for ungapped q-grams, and it is not possible anymore to use a suffix array for the gapped Q-grams. Instead, the database is scanned twice. The first time the number of occurrences of all Q-grams is counted.

In the second scan, the positions at which a q-gram starts are recorded in an array of size n. During that scan, the index points to the start of the respective list.

The detail shall be worked out as an *exercise*.

5.12 Extension to Levenshtein distance

Note that the q-gram method presented so far can only be used to find local approximate matches with the *Hamming* distance.

The q-gram method can be generalized to the *Levenshtein* distance. Burkhardt and Kärkkäinen have described an extension that uses 'one-gapped q-grams'.

The idea is to model insertions and deletions by additional Q-grams. For example, with the basic shape ##-# applied the text, we would use ##-#, ##--#, and ### for the pattern.

The filter then compares all three shapes in the pattern to the q-grams of the basic shape in the text. Thus matching q-grams are even found in the presence of indels.

Otherwise he algorithm stays essentially unchanged, except that the threshold computation is slightly different.

5.13 Summary

- Filtering based on q-grams using a suffix array with an index is an efficient filtering method.
- In the gapless case, filtering efficiencies of $\approx 0.2\%$ were observed for genomic sequences.
- Gapped Q-grams improve the filtering efficency further (by orders of magnitude).
- The threshold t and the minimum coverage both influence the filter efficency.
- No closed formula is known for computing t for gapped Q-grams.