HAL
open science

YASS: Similarity search in DNA sequences

Laurent Noé, Gregory Kucherov

To cite this version:

Laurent Noé, Gregory Kucherov. YASS: Similarity search in DNA sequences. [Research Report] RR-4852, INRIA. 2003, pp.20. inria-00071731

HAL Id: inria-00071731
https://inria.hal.science/inria-00071731v1

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

YASS: Similarity search in DNA sequences

Laurent Noé — Gregory Kucherov

N° 4852

Juin 2003

THÈME 2

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4852--FR+ENG

YASS: Similarity search in DNA sequences

Laurent Noé , Gregory Kucherov

Thème 2 — Génie logiciel
et calcul symbolique
Projet Adage

Rapport de recherche n° 4852 — Juin 2003 — 20 pages

Abstract: We describe `YASS` – a new tool for finding local similarities in DNA sequences. The `YASS` algorithm first scans the sequence(s) and creates on the fly groups of *seeds* (small exact repeats obtained by hashing) according to statistically-founded criteria. Then it tries to extend those groups into similarity regions on the basis of a new extension criterion.

The method can be seen as a compromise between single-seed (`BLAST`) and multiple-seed (`FASTA`, `BLAT`) approaches, and achieves a gain in both sensitivity and selectivity. The method is flexible and can be made more efficient by using spaced seeds, and in particular transition-constrained spaced seeds.

We provide examples of applying `YASS` to *Saccharomyces Cerevisiae* and *Drosophila Melanogaster* chromosomes.

Key-words: YASS, local alignment, spaced seeds, transitions

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

YASS: Recherche de similaritées dans les séquences d'ADN

Résumé : Nous présentons `YASS` – un nouvel outil par la recherche locale de similaritées dans les séquences d'ADN. L'algorithme de `YASS` parcours la séquence dans un premier temps, et crée des groupes de graines (petites répétitions exactes obtenues par hachage) selon des critères reposant sur des propriétées statistiques. Dans un deuxième temps, il essaie d'étendre ces groupes en régions de similaritées selon un nouveau critère d'extension.

La methode proposée peut être vue commme un compromis entre les stratégies à une seule graine (`BLAST`) et celles à multiples graines (`FASTA`, `BLAT`), elle atteind des gains à la fois sur la sensibilitée et la selectivité. La méthode reste flexible et peut être rendue encore plus efficace en utilisant des graines espacées, particulièrement en considérant des graines espacées contenant des elements spécifiques contraints aux transitions.

Nous donnons des examples d'utilisation de `YASS` sur des chromosomes de *Saccharomyces Cerevisiae* et *Drosophila Melanogaster*.

Mots-clés : YASS, alignement local, graines espacées, transitions

1 Introduction

The well-known Smith-Waterman algorithm [20] provides an exact solution to the problem of computing optimal local alignments. However, its time complexity is high ($\mathcal{O}(n^2)$, reducible to $\mathcal{O}(\frac{h.n^2}{\log n})$ where h is the sequence entropy [8]), and rapid heuristic algorithms have been proposed to find significant approximately repeated sequences (similarity regions). All those algorithms are based on a common principle: at the first stage, they search for one or more small exact repeats (called *seeds*) assumed to occur in approximate repeats. The second stage attempts to extend those seeds to compute longer approximate repeats.

Typical representatives of this approach are `FASTA` [14, 15] and `BLAST` [2], that both start with identifying, using hash methods, small exact matches of fixed size. However, they differ in the way they treat those seeds to extend them into similarity regions. `FASTA` looks for groups of closely-located seeds *on the same dot plot diagonal* trying to identify a longer similarity region. In contrast, `BLAST` systematically tries to extend each *seed* (considered to be a *hit*) to a possible HSP (High Scoring Pair), using an *X-drop* algorithm. A modified *double-seed* criterion has been proposed in `Gapped BLAST` [3], that defines a hit as two non-overlapping seeds occurring on the same diagonal.

Different extensions of these two methods have been proposed by other algorithms. `ASSIRC` [23] applies a random walk heuristics to perform the extension phase. Another generalization concerns the form of *seeds*: `FLASH` [7] and more recently `PatternHunter` [16] consider *spaced seeds*, that require matches to occur at certain positions within a given length, according to a pre-specified *seed motif*. Double-seed and spaced seeds approaches have been combined in `BLASTZ` [19, 18].

Other methods try to take advantage of powerful string matching techniques, in particular by using a suffix tree data structure and a linear algorithm for constructing it [22]. This is done in `REPuter` [13] and `MUMmer` [9], to construct respectively local or global alignments.

Other tools have been designed to compute special type of alignments: `MEGABLAST` [24] or `SSAHA` [17] are specialized in finding large and highly similar regions; `SIM4` [10] or `BLAT` [11] are oriented to computing *spliced alignments*, useful in mapping RNA transcripts on the genomic DNA; finally, `BLASTX` [21] is designed to match DNA queries against a protein database.

Other related tools have the goal of finding *tandem repeats*. One such software is `Tandem Repeat Finder` [4], that is also based on collecting seeds with a hashing technique, using a group criterion adapted to identifying tandemly-repeated sequences. Some ideas of this approach will be used in this paper. `mreps` [12] is another program for finding tandem repeats with substitution errors, based on efficient combinatorial algorithms.

Our Approach

The two-stage approach described above implies a dilemma between the sensitivity and selectivity. Sensitivity can be measured by the probability that a good alignment (one with a high score) is found by the algorithm. Selectivity can be associated with the number of

candidate *hits* computed at the first stage that don't yield significant approximate repeats at the extension stage (smaller this number, higher the selectivity). Choosing weaker criteria for defining a hit (e.g. decreasing the seed size) increases the sensitivity but makes the algorithm less selective and increases the time spent on extending spurious hits. To cope with this problem and to increase the efficiency, some algorithms try to speed up the extension stage by using an heuristics [23], others introduce an additional treatment to rapidly filter out those hits that are unlikely to produce a significant local alignment [18]. On the other hand, choosing a more stringent criteria for defining a hit (e.g. increasing the seed size) makes the first stage more selective, saves time, but can miss significant similarities and therefore results in a less sensitive algorithm.

Therefore, finding an optimal trade-off between the sensitivity and selectivity while keeping the efficiency is a key issue for the entire approach. For example, the double-seed method of `Gapped BLAST` has been shown more to be sensitive than `BLAST` for HSPs with a bit-score of at least 33 bits (for the default seed size 13 for `BLASTN` and 11 for `Gapped BLAST`), but also more selective, as less time is spent on treating randomly occurring hits.

In this work, we generalize this approach and propose an algorithm based on a flexible *multiple-seed criterion* that allows an *arbitrary number of possibly overlapping seeds.* Moreover, seeds are not required to occur on the same dot plot diagonal. In other words, we admit a variation of the distance between corresponding seeds in each copy, which allows us to account for indels occurring in inter-seed intervals.

Considering multiple seeds and trying to group them has the advantage of catching the whole range of a similarity region rather than spotting a single part of it. For the sensitivity reasons, we use a smaller seed size. This implies generating more seeds at the first stage, but only a small fraction of them form groups verifying the extension criterion and is therefore processed by the extension step.

Overall, the sensitivity is increased due to a smaller seed size and the flexibility of the extension criterion, and the selectivity is preserved as the criterion requires the presence of several seeds to trigger the extension. The criterion is easy to verify, and the extra time spent on computing smaller seeds at the first stage is greatly compensated by the gain in time spent on computing similarity regions from the seeds.

The paper is organized as follows. Section 2 presents an overview of the `YASS` algorithm. Section 3 defines the criteria used to group seeds, and describes the grouping algorithm. Section 4 deals with the extension criterion and sensitivity improvements. Section 5 concerns spaced seeds and model improvements. Sections 6 describe results and experiments, as well as possible optimizations of the method.

2 Algorithm Overview

The initial step of the algorithm consists of collecting the information about all possible seeds contained in the input sequence(s). This is done in the traditional manner: given a size k, we store in a hash table the positions of all k-words occurring in the sequence. For each k-word, the hash table contains a linked list of its positions in the sequence.

After this preparatory step, the algorithm is composed of two parts. The first part is a *linking algorithm.* It considers *seeds*, i.e. repeated k-words extracted from the hash table, and processes them to form *groups of seeds*, according to criteria based on the distances between corresponding k-words.

The second part is an *extension algorithm* that triggers and performs the extension of some of the constructed groups of seeds. Triggering the extension is driven by a selection criterion, called *group criterion*, based on the *total nucleotide size of the group* (note that seeds can overlap). Groups of seeds verifying this criterion are actually submitted for extension. In contrast to the *X-drop* algorithm of `BLAST`, this criterion is very easy to check and virtually no time is spent on verifying it. Selected groups are then extended to form similarity regions. For that, respective inter-seed regions are aligned using a dynamic programming algorithm. Besides, an additional heuristic procedure is applied in order to check whether the region consists actually of one or several sub-regions of higher score. This is done in order to correct the artifact of the group criterion that consists of possible binding of closely-located high-scoring regions that "shadow" a low-scoring region in between.

Linking and extension algorithms are run in turn in order to keep the currently stored set of groups small enough.

3 Linking Algorithm

We first introduce some notations used in this part. We are looking for local similarities between a DNA query sequence $Q[1..m]$ and a DNA subject sequence $T[1..n]$. Each of those sequences can be considered as a succession of $m-k+1$ (respectively $n-k+1$) substrings of size k, called *k-words.* Two matching k-words $Q[j..j+k-1]$ and $T[i..i+k-1]$ form a *seed*, denoted $<i,j>$. Two distances between k-words are considered:

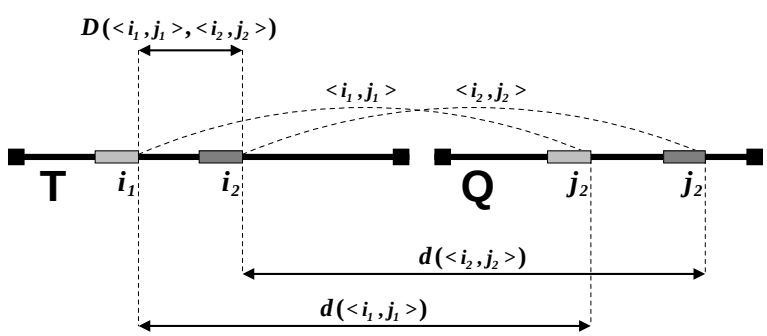

Figure 1: *intra-seed and inter-seed distances*

- Given a seed $<i,j>$, the *intra-seed* distance $d(<i,j>)$ is $n+j-i$. It can be seen as the distance between the k-words $Q[j..j+k-1]$ and $T[i..i+k-1]$ if Q is concatenated to T (see Figure 1),
- Given two seeds $<i_1,j_1>$ and $<i_2,j_2>$, where $i_1 < i_2$ and $j_1 < j_2$, the *inter-seed* distance $D(<i_1,j_1>,<i_2,j_2>)$ is the maximum between $i_2 - i_1$ and $j_2 - j_1$.

Two *seeds* $<i_1,j_1>$ and $<i_2,j_2>$ are linked (grouped) together if

- the *inter-seed* distance is below a threshold ρ ($D(<i_1,j_1>,<i_2,j_2>) \leq \rho$),
- the variation between the two *intra-seed* distances $d(<i_1,j_1>)$ and $d(<i_2,j_2>)$ is below a threshold δ ($|d(<i_1,j_1>) - d(<i_2,j_2>)| \leq \delta$).

The first inter-seed condition insures that the seeds are close enough to each other. The second intra-seed condition requires that in both seeds the two k-words occur *approximately* at the same distance from each other. In other words, the two seeds are located at *close diagonals*. Allowing a variation between intra-seed distances accounts for indels that might have occurred between the corresponding k-words inside a similarity region (see Figure 1).

Parameters ρ and δ, used in the above conditions, are estimated according to statistical models of the DNA sequence that we describe now.

Statistical models

Substitutions. Two similar DNA regions are assumed to stem from a duplication of a common ancestor sequence, followed by individual independent substitution events occurring in each duplicated copy. Under this assumption, the two regions have an equal length and their alignment is a sequence of matched and mismatched pairs of nucleotides. We then model this alignment by a sequence of i.i.d Bernoulli variables, with the probability p for a match and $(1-p)$ for a mismatch. To estimate the inter-seed distance, we have to estimate the distance between two *successive runs of at least k matches* in the Bernoulli sequence. It obeys the geometric distribution of order k called the *Waiting time distribution* [1]:

$$\mathcal{P}\{D_k = x\} = \begin{cases} 0 & \text{for } 0 \leq x < k \\ p^k & \text{for } x = k \\ (1-p)p^k[1-\sum_{i=0}^{x-k-1}\mathcal{P}\{D_k = i\}] & \text{for } x > k \end{cases}$$

Using this distribution, we define ρ such that the probability $\mathcal{P}\{D_k \leq \rho\}$ is high (the default value is 0.95 but can be changed by the user).

Note that the Waiting time distribution allows us to estimate another useful parameter: the number of runs of matches of size at least k inside a Bernoulli sequence of length x. In a Bernoulli sequence of length x, the probability of the event $I_{p,x,r}$ of having exactly r *non-overlapping* runs of matches of size at least k is given by the following recursive formula:

$$\mathcal{P}\{I_{p,x,r}\} = \mathcal{P}\{I_{p,x-1,r}\} + p^k(1-p)\big(\mathcal{P}\{I_{p,x-k-1,r-1}\} - \mathcal{P}\{I_{p,x-k-1,r}\}\big)$$

This gives the probability of having exactly r *non-overlapping* seeds of size at least k inside a repeat of size x. The recurrence starts with the case $r = 0$ in which case $\mathcal{P}\{I_{p,x,0}\} = \mathcal{P}\{D_k > x - k\}$ and is computed through the Waiting time distribution.

The distribution $\mathcal{P}\{I_{p,x,r}\}$ allows to infer a lower bound on the number of non-overlapping seeds expected to be found inside a similarity region. In particular, we will use this bound as a first estimate of the group criterion introduced later in Section 4.

Indels. *Indels* (insertion/deletion of nucleotides) are responsible for a diagonal shift of seeds viewed on a dot plot alignment. In other words, they modify the relation $d(<i_1, j_1>) = d(<i_2, j_2>)$ by adding a possible (small) bias. To estimate a typical shift size, we use a method similar to one proposed in [4] for tandem repeats.

The method assumes that a single nucleotide indel can occur with an equal probability q at each of l nucleotides separating two consecutive seeds. Under this assumption, estimating the diagonal shift produced by indels is done through a discrete one-dimensional *random walk* model, where the probability of moving left or right is equal to q, and the probability of staying in place is $1 - 2q$. Our goal is to bound, with a given probability, the deviation from the starting point.

The probability of ending the random walk at position i after l steps is given by the following sum:

$$\sum_{j=0}^{(l-i)/2} \frac{l!}{j!(j+i)!(l-(2j+i))!} p_i^{2j+i}(1-2p_i)^{l-(2j+i)}$$

A direct computation of multi-monomial coefficients quickly leads to a memory overflow, and to avoid this, we used an indirect method based on generating functions. Consider the function $P(x) = qx + (1-2q) + \frac{q}{x}$ and consider the power $P^l(x) = a_l.x^l + \cdots + a_{-l}.x^{-l}$. Then the coefficient a_i computes precisely the above formula, and therefore gives the probability of ending the random walk at position i after l steps. We then have to sum up coefficients a_i for $i = 0, 1, -1, 2, -2, \ldots, l, -l$ until we reach a given threshold probability (0.95 by default). The obtained value l is then taken as the parameter δ used to bound the maximal diagonal shift between two seeds.

Linking Algorithm

We now describe our linking algorithm that processes seeds on the fly and creates groups according to the inter- and intra-seed criteria described above. The algorithm is given on Figure 2.

Along the generation of seeds, we store each seed $<i, j>$ in a table D according to its intra-seed distance $d(<i, j>)$ (actually, only position i is stored). Verifying if a newly generated seed should be linked with previously computed ones is done in the following way:

Figure 2: *Linking algorithm*

```
01  forall k-word w = T[i..i+k-1] (1 ≤ i ≤ n-k+1) do
02     forall j such that T[j..j+k-1] matches w do
03        d ← i - j
04        for d_obs ∈ [d, d+1, d-1, ..., d+δ, d-δ] do
05           i' ← D[d_obs]
06           if i - i' < ρ then
07              j' ← i' - d_obs
08              group together <i', j'> and <i, j>
09              break //the d_obs loop
10           endif
11        endfor
13        D[d] ← i
13     endforall
14  endforall
```

- First, we look for previously computed seeds verifying the intra-seed criterion (line 04). For that, we retrieve from table D the seeds with an intra-seed distance belonging to the interval
  $[d(<i,j>) - \delta..d(<i,j>) + \delta]$, in the increasing order of its difference from d.
- From these candidate seeds, we select the first one which satisfies the inter-seed condition (line 06).

A group is completed if its last linked seed has an inter-seed distance of more than $\rho + \delta$ from the current seed.

4 Extension algorithm

The linking algorithm presented in the previous section forms groups of seeds potentially belonging to the same similarity region. Among those groups, groups containing a large number of seeds with small intra-seed distances represent large and high-scored similarities. However, many groups will contain a smaller number of seeds, possibly separated by larger inter-seed distances. Those groups either correspond to similarities with a lower score, or do not correspond to any similarity at all (i.e. don't belong to an alignment with a sufficiently big score). How can we distinguish between those two cases, without performing an alignment algorithm, and without missing many similarities with a low but sufficient score?

For this purpose, we introduce another criterion. This criterion selects groups that will be actually extended. The criterion, called *group criterion*, is based on the *group size* – the

overall nucleotide size of all seeds of the group. Note that seeds of a group can overlap, and therefore the group size is not just the number of seeds times k, but can be generally smaller.

Allowing overlapping seeds is an important point that brings a flexibility to our method. Note that other popular multiple-seed methods (`Gapped BLAST`, `BLAT`) consider only non-overlapped seeds fixing their minimal number to two. Overlapped seeds allow a trade-off between those two options, increasing the sensitivity of the usual multiple-seed approach (with respect to low-scoring similarities) without provoking a tangible increase in the number of useless extensions. In the next section, we will provide quantitative measures comparing the sensitivity of the `YASS` group criteria with `BLAST` and `Gapped BLAST`.

Note that `FASTA` and `BLAST` methods are somewhat opposite, as each one spends most of its time doing what the other does quickly. `FASTA` spends most of the time on generating and counting potential seeds along the same diagonal (extension stage is strait-forward), whereas `BLASTN` concentrates on extending *each* seed (seed generation is comparatively fast). Figure 3 shows that `YASS` reaches a trade-off between those two approaches, as it spends approximately equal time for each of the two stages. This can be seen as an evidence that `YASS` provides an optimal distribution between the two complementary parts of the work. Figure 3 focuses on comparing chromosomes V and IX of *S.Cerevisiae*, and shows the running times for different seed size and group size, together with corresponding values of δ and ρ parameters used in the linking algorithm. All experiments reported in this work have been done on a Pentium IV 2 GHz computer.

Figure 3: *Comparing* S.Cerevisiae *chromosome V (576 869 bp) vs chromosome IX (439 885 bp).* t_{link}, t_{align} *and* t_{total} *indicate the time spent on respectively the linking algorithm, extension algorithm and both.*

size				**cpu time**		
seed	*group*	δ	ρ	t_{link}	t_{align}	t_{total}
9	13	135	5	2s	3s	**5s**
9	11	135	5	2s	6s	**8s**
8	13	97	4	7s	6s	**13s**
8	11	97	4	7s	11s	**18s**
7	13	69	4	22s	35s	**47s**
7	11	69	4	22s	41s	**63s**

Comparison of methods

Similarity of two DNA regions is traditionally measured by the alignment score (or similarity score) under some scoring system. Unless otherwise stated, we use the standard `BLAST` scoring system: +1 for a match and -3 for a mismatch. Choosing another scoring system does not change the nature of the results. Furthermore, in this section we consider gapless alignements only. If we fix a score of such an alignement, we can infer a dependency between its length and the identity rate (fraction of matched characters). Figure 4 shows the relation

for some typical score value. It illustrates that for a fixed score, short similarities will match

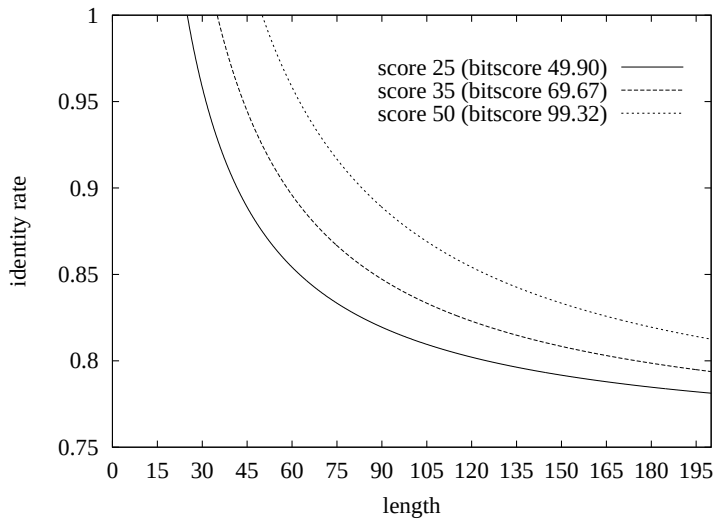

Figure 4: *Dependency between length and identity rate of gapless fixed-score alignments*

very well while long ones will have the identity rate tending to 75% (depending on the scoring system).

To estimate the sensitivity of a similarity search method, we fix a score and measure the probability of hitting a similarity of a certain length, *among all similarities of this length and of the given score.* Furthermore, the probability is measured on *maximal scoring sequences.* Those are sequences such that none of its substrings reaches a greater score.

In particular, we measure the probability of finding one seed of size 11 (default `BLAST` single-seed criterion) in comparison to multiple-seed search. Searching for two seeds of size 9 using the double-seed criterion of `Gapped BLAST` in similarity regions scoring 25 is more sensitive if the sequences are long (see Figure 5). On the other hand, the single-seed `BLAST` strategy works better for short (and therefore highly identical) sequences.

We now apply our group criterion with seed size 9 (same as for `Gapped BLAST`) and group size 13. This is less selective than the `Gapped BLAST` criterion, as it includes any two non-overlapping seeds but also includes pairs of overlapping seeds with an overlap at most 5. According to Figure 5, the method outperforms both methods above: it is more sensitive than the single-seed (size 11) criterion even for short similarities, and than the non-overlapping double-seed (size 9) criterion for large similarities.

To estimate the selectivity of different methods, we compute the probability of a hit at a given position in a random i.i.d. Bernoulli sequence (see [11]). For our approach, this probability is $2.1 \cdot 10^{-8}$, which improves the one of `BLAST` ($2.4 \cdot 10^{-7}$) by more than ten.

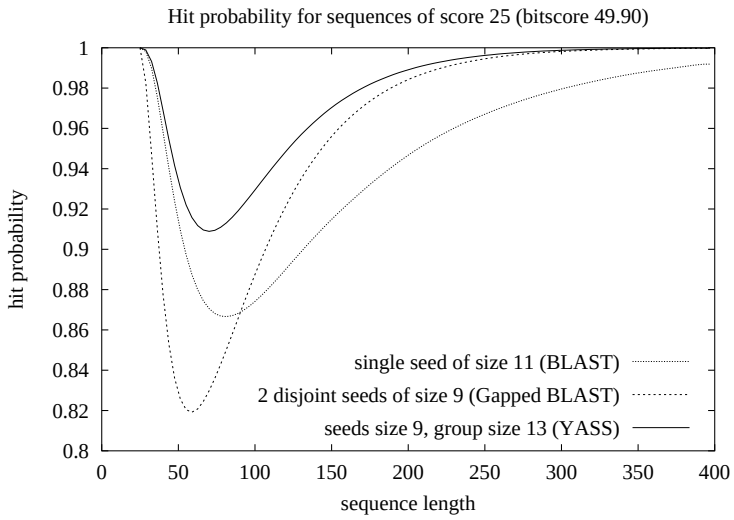

Figure 5: *Hit probability as a function of length of fixed-score alignements*

In `Gapped BLAST`, this probability ($7.28 \cdot 10^{-9}$) is smaller than the one for our method, as the `Gapped BLAST` double-seed criterion is strictly stronger than ours. On the other hand, `Gapped BLAST` is much less sensitive on short sequences.

To be able to still compare `Gapped BLAST` with our approach, we chose a parameter configuration such that both algorithms have the same selectivity (10^{-6}). This is achieved by choosing seed size 8 for `Gapped BLAST` and group size 11 for `YASS` (while keeping seed size 9).

In this setting, and for sequences of a fixed score (25 in our experiment), `YASS` turns out to be more sensitive for sequences up to 80 bp, while `Gapped BLAST` becomes more sensitive for longer sequences (data not shown). Also, `YASS` is more efficient in this case, as `Gapped BLAST` tends to compute more spurious individual seeds that are not followed by a second hit, which takes a considerable part of the execution time. On the other hand, since the `YASS` seed size is smaller by one nucleotide, the number of spurious individual seeds at the first step is then divided by 4 on large sequences.

Let us now analyze the comparative behavior of the `YASS` group criterion for other score values. Figure 6 shows the hit probability of the group criterion (seed size 9, group size 13) and the single-seed criterion of `BLAST` (seed size 11). Figure 6 shows that for alignments of score larger than 25 (that are generally of interest for DNA sequences), the `YASS` criterion is more sensitive than that of `BLAST` *for all alignment lengths.* If the score becomes smaller, the group criterion becomes less sensible for shorter sequences but is still more sensitive for larger ones (score 20 on Figure 6). However, for such a small score the sensitivity of both

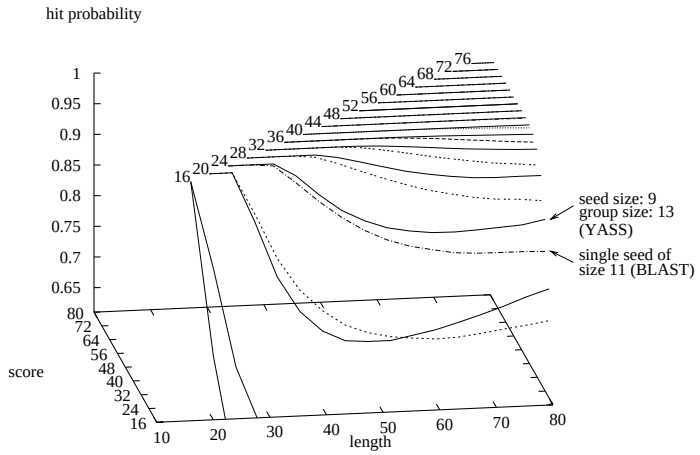

Figure 6: *Sensibility of single-seed (*`BLAST`*) vs group criterion (*`YASS`*) for different score values*

methods is weak (under 0.8). For yet smaller scores (score 16 on Figure 6), both methods fail except for very short sequences, in which case the single-seed method turns out to be preferable. In practice, detecting so low-scoring similarities requires reducing the seed size.

5 Spaced seeds

So far we have assumed that nucleotide mutations occur independently along similarity regions. However, this assumption is not always justified. In particular, in protein-coding regions, the third codon base is more prone to mutations than the first and the second one.

A way to take this observation into account is to use *spaced seeds*. In contrast to classical seeds that correspond to k contiguous nucleotide matches, spaced seeds are represented by a *shape* which specifies positions at which matches must occur. For example, `#.##` specifies that two 4-words form a seed, if matches occur at the first, third and fourth positions but not necessarily at the second one. A seed is characterized by its *weight* w (number of `#`) and its *span* l (total length), and is completely defined by its shape. The choice of the shape is important and directly affects the efficiency of the seed.

Spaced seeds have been shown to considerably improve the sensitivity, not only on protein-coding regions but also on general unconstraint DNA sequences. Spaced seeds have been designed and systematically used in `PatternHunter` [16], and have also been studied, from a more theoretical perspective, in [6]. The recent work [5] proposed a tool, called

`Mandala`, for designing most sensitive spaced seeds according to a mutational model specified by a boolean Markov chain. `YASS` is compatible with the spaced seed approach and

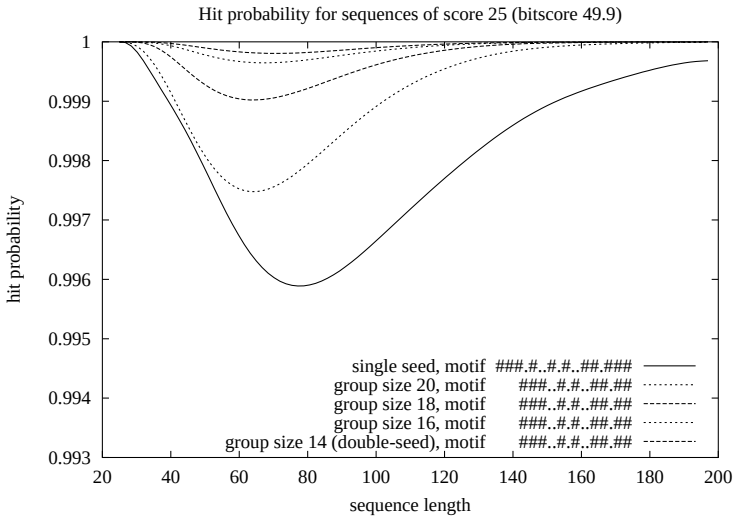

Figure 7: *Hit probability of spaced seeds. The solid line represents the hit probability for a single seed of weight 11 (implemented in* `PatternHunter`*). The other lines correspond to a seed of weight 9, and the* `YASS` *group criterion, for different group sizes. Note that any overlap of this seed with itself yields at least 14 matched nucleotides, and therefore setting the group size to 14 amounts to the double-seed criterion of* `PatternHunter`

allows the user to propose his own seed shape. Note that applying the group criterion in the case of overlapping spaced seeds is less trivial, as the total number of matches depends not only on the relative placement of the seeds but also on the shape itself. To count this number on-line, `YASS` uses an appropriate finite automaton computed from the seed shape.

`PatternHunter` combines spaced seeds with the double-seed criterion, allowing a possible overlap between the two seeds. The approach is very efficient in both selectivity and sensitivity, in comparison to methods based on contiguous seeds. However, its efficiency depends on the choice of the shape, as it determines the possible number of matched nucleotides in overlapped seeds (which, in turn, directly affects the selectivity). For example, possible self-overlaps of the seed `###..#.#..##.##` (of weight 9) have the number of matched nucleotides varying from 14 to 18. Spaced seeds designed for aligning coding regions usually have a regular structure (typically, every third position is a space). On the other hand, unrestricted self-overlaps of such regular motifs may result in a large interval of the number of matches, including low values which can provoke more infertile hits and therefore worsen the selectivity.

Figure 8: *Similarity region in chromosomes V vs IX of* S.cerevisiae. *Running* `BLAST` *takes 4s with seed size 11 (default) and over 30s with seed size 10. The region shown is found by* `BLAST` *with seed size 10 only, as it does not contain 11 contiguous matches. It takes* `YASS` *13s to identify this region (seed size 8, group size 13), and this time can be decreased to 6s using spaced seeds adapted to gene comparison (see Section 5).*

```
*(270758-271079)(257865-258186) score:27  Ev: 0.000240824  s: 322/322 f
* V.fas (forward strand) / IX.fas (forward strand)
* mutations per triplet 9, 18, 58 (1.44e-12) | ts : 45 tv : 40

  |270760   |270770   |270780   |270790   |270800   |270810   |270820   |270830   |270840   |270850   |270860   |270870   |270880   |270890
CTTCCATTTCCATGTCAACGCCTAATGGCAATGCTGCAACAGCAACGCATGATCTGGCAGGTTTGTGAGTATTGAAGTACTTGGCGTAAACGGAGTTAAACTCAGCAAAGTGATTGATATCTGCCAAGAAAATGTTAACT
||||||||||.|:.|||||....||:||.||:|::|||||||:|||||.||:||||||:||||||.||.||.||.|:||||||..||||||||:||.||||||.||.||.|||||||||::||:||.||:||||||||:|.||||.|||
CTTCCATTTCTAAATCAACATTCAAAGGTAAGGAAGCAACACCAACACAGGATCTTGCAGGCTTATGGGTGTGGAAGTGTTTGGCGTATACAGAGTTGAATTCGGCAAAGTTTTTCATGTCAGCCAAGAATACGTTGACT
     |257870   |257880   |257890   |257900   |257910   |257920   |257930   |257940   |257950   |257960   |257970   |257980   |257990

  |270900   |270910   |270920   |270930   |270940   |270950   |270960   |270970   |270980   |270990   |271000   |271010   |271020   |271030
TTTACGACCCTGTCCAATGAGGAATTGCTTGCTTCTAGAACATTCTTAATGTTTTGAATCACCTGTTCAGCCTTATCAGCAATGGAACCTTCAACTAACTTGTTGTCTGGGGTCACTGGAATTTGGCCAGAAAGGAAAAT
||:||:|.:.||||.||:||.||||||.|||:|||:|||..|.||||||||.||||||||::||.||||||.||||||:||||:.||.||||||||:|||:..||||||.||||||.||:::||.||||||.||||||:|:|:|:|.
TTGACTATATTGTCTAAAGAAGAATTACTTTCTGCTAAGATATTCTTAACGTTTTGAAAAACTTGTTCGGCCTTCTCAGAGATAGAACCTTGAACAGGCTTGTTATCTGGAGTATAAGGGATTTGACCAGACACGTACAC
     |258010   |258020   |258030   |258040   |258050   |258060   |258070   |258080   |258090   |258100   |258110   |258120   |258130

  |271040   |271050   |271060   |271070
CAAATTGTTCACTTTCATGGCATGGGAGTATGAAGCCGCA
:||||||||:.|.||||||.||:||||||||:||.||:|||
AAAATTGTTGGCCTTCATAGCTTGGGAGTAAGAGGCGGCA
     |258150   |258160   |258170
```

Figure 9: *Protein sequences corresponding to cDNAs of Figure 8*

```
                MVTTLTPVICESAPAAAASYSHAMKVNNLIFLSGQIPVTPDNKLVEGSIADKAEQVIQNIKNVLEASNSSLDRVVKVNIFLADINHFAEFNSVYAKYFNTHKPARSCVAVAALPLGVDMEMEAIAAERD*
                 +TTLTPV  + APPAAASYSQAMK NN +++SGQIP TPDNK V+GSI++KAEQV QN+KN+L  SNSSLD +VKVN+FLAD+ +FAEFNSVYAK+F+THKPARSCV VA+LPL VD+EME IA E++
MFLRNSVLRTAPVLRRGITTLTPVSTKLAPPAAASYSQAMKANNFVYVSGQIPYTPDNKPVQGSISEKAEQVFQNVKNILAESNSSLDNIVKVNVFLADMKNFAEFNSVYAKHFHTHKPARSCVGVASLPLNVDLEMEVIAVEKN*
```

In this context, the `YASS` group criterion provides an advantage of suppressing hits obtained by a small number of matches. This can in general improve the selectivity without sacrificing the sensitivity. Figure 7 shows a comparisons of the sensitivity of spaced seeds for different group sizes. In particular, setting the group size to 16 yields the sensitivity close in practice to the one for group size 14 (`PatternHunter` case) – both allow to detect more than 99.9% of alignements scoring 25. On the other hand, setting the group size to 16 greatly improves the selectivity, as the number of infertile hits is roughly divided by 16. Another interesting fact shown on Figure 7 is that even for the group size as large as 20, the `YASS` method is still more sensitive than the single-seed method, when only 11 nucleotides are required to match.

An improvement proposed by `YASS` here is the possibility to specify spaced seed positions that admit only transitions (mutations `A` $\leftrightarrow$ `G`, `C` $\leftrightarrow$ `T`) rather than any mutations. This allows to improve the sensitivity/selectivity ratio, as transitions are known to occur more frequently in DNA sequences than transversions – in particular, transitions occurring at the

third codon base don't change the resulting amino-acid in most of the cases. Introducing this feature led us to modify the definition of the weight of a seed and to consider the *bit-weight* defined as $card(\#) + 0.5 \cdot card(@)$.

Table 1: *Effect of seed size and motif illustrated on* Drosophila Melanogaster *chromosome II (20 302 692 bp).* $\#_{seeds}$ *is the number of individual seed occurrences,* $\#_{groups}$ *the number of obtained groups,* $\#_{alignments}$ *the number of groups submitted to extension (after applying the group size criterion), and* $\#_{results}$ *the number of resulting alignments*

weight			cpu time			results			
seed	*motif*	*grp*	t_{link}	t_{align}	t_{total}	$\#_{seeds}$	$\#_{groups}$	$\#_{alignments}$	$\#_{results}$
11	contiguous	20	239s	291s	530s	477621164	99229643	3552479	72245
11	###.#..#.#..##.###	20	101s	71s	172s	104861618	2199996	1434228	201438
11	#@##@##...@##@##	20	109s	62s	171s	108420076	2472847	974097	219547
9	contiguous	20	778s	361s	1139s	1656484859	342604276	4473824	518852
9	###..#.#..##.##	20	615s	225s	840s	1329158627	36719691	5635528	534555
9	@#@##@#.#.@##	20	651s	111s	762s	1225490772	48722375	3380259	536310

As an illustration, Table 1 shows comparative results for a contiguous seed, a regular spaced seed, and a spaced seed with transition constraints, of the same bit-weight (11 and 9). The results shown have been obtained on *Drosophila Melanogaster* chromosome II. For each of the seeds, `YASS` was applied with the group size 20. The table illustrates that spaced seeds improve both the sensitivity and the selectivity considerably over contiguous seeds. The selectivity can be associated with the number of performed alignements (column $\#_{alignements}$) and the sensitivity with the number of significant similarities found ($\#_{results}$). On the other hand, introducing transition constraints allows an additional increment in both the selectivity and the sensitivity, as well as a gain in the execution time. It is interesting to note that the number of occurrences of individual seeds ($\#_{seeds}$) is smaller for the transition-constrained seed than for the regular spaced seed, while it is inverse for the number of groups formed by those seeds ($\#_{groups}$). This can be interpreted by that occurrences of transition-constrained seeds tend to be better grouped and therefore are more suitable to the group criterion we use. Furthermore, the number of performed alignments ($\#_{alignements}$) is again smaller for transition-constrained seeds, which means that the group criterion does a very good job by cutting out a big part of candidate groups.

6 Results and Experiments

Program

The method described in the previous sections has been implemented in `YASS` (*Yet Another Similarity Searcher*) software.

The executable code is available at `http://www.loria.fr/~noe/`, currently for Sun Solaris 5, dUnix on DEC Alpha, Linux on i386 and Microsoft Windows. Detailed results of running `YASS` on largest *S.Cerevisiae* chromosomes are given in Table 2.

Table 2: *Comparisons of* S.Cerevisiae *chromosomes obtained with seed size 9 and group size 13.*

sequence		size		cpu time			results			
Query	*Text*	m	n	t_{link}	t_{align}	t_{total}	$\#_{seeds}$	$\#_{groups}$	$\#_{align.}$	$\#_{results}$
IV	IV	1531912	1531912	13s	16s	29s	27835943	5575829	334732	37807
IX	IV	439885	1531912	5s	6s	11s	10231800	2060951	123657	13829
V	IV	576869	1531912	7s	9s	16s	13665079	2735724	165224	19508
XVI	IV	948061	1531912	12s	14s	26s	22758709	4562398	274390	30988
IX	IX	439885	439885	1s	2s	3s	2156446	430731	27016	3434
V	IX	576869	439885	2s	2s	4s	3787759	759558	46546	5439
XVI	IX	948061	439885	3s	3s	6s	6283229	1262067	76097	8589
V	V	576869	576869	2s	2s	4s	3772422	752608	46338	5822
XVI	V	948061	576869	4s	5s	9s	8394013	1676487	101319	11945
XVI	XVI	948061	948061	5s	6s	11s	10469937	2099612	127230	14456

Further options and example

As already mentioned before, transitions occur generally more frequently than transversions. In particular, transition mutations of the codon third base of coding regions don't change the resulting amino acid most of the time. For this reason, `YASS` has an option of changing the scoring system and assigning a smaller penalty to transition substitutions than to transversions. Using this option, `YASS` was able to identify a similarity region on chromosomes V vs IX of *S.Cerevisiae*, shown in Figure 8. This region is not found by `BLAST` (even with corresponding modifications of the scoring system), since it does not contain a contiguous seed of size 11. In this example, a clear bias is observed in mutations at each of the three positions of the reading frame (58 mismatches of the third nucleotide, and only 9 and 18 mismatches of respectively the first and the second nucleotide). This suggests that the found regions of similarity are coding. A closer examination of the annotation of *S.Cerevisiae* (`http://pedant.gsf.de/`) reveals that they are coding for the following proteins:

- HMF1 — Heat-shock induce-able Inhibiter of cell Growth hypothetical protein.
- MMF1 — required for maintenance of mitochondrial DNA hypothetical protein.

Those two homologous proteins belong to the same super-family HI0719, and have a high sequence similarity (see Figure 9).

Conclusion

We have presented an efficient strategy for finding similarity regions in DNA sequences, implemented in `YASS` software. The method is based on a multiple-seed hit criterion, allowing overlapping seeds. Occurring seeds are linked into groups, and this process is governed by parameters automatically computed according to statistical models. A new group criterion is then applied to select groups which are submitted to the extension stage. The method can be combined with the spaced seed approach, reaching an additional gain in sensitivity.

Acknowledgements

Authors are grateful to Marie-Pierre Etienne, Roman Kolpakov, Gilles Schaeffer and Pierre Valois for their help. This work has been supported by the french *Action Spécifique "Algorithmes et Séquences"* of CNRS.

References

[1] Aki, S., Kuboki, H., and Hirano, K. On discrete distributions of order k. *Annals of the Institute of Statistical Mathematics 36* (1984), 431–440.

[2] Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. Basic Local Alignment Search Tool. *Journal of Molecular Biology 215* (1990), 403–410.

[3] Altschul, S., Madden, T., Schäffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research 25*, 17 (1997), 3389–3402.

[4] Benson, G. Tandem repeats finder: a program to analyse DNA sequences. *Nucleic Acids Research 27*, 2 (1999), 573–580.

[5] Bulher, J., Keich, U., and Sun, Y. Designing seeds for similarity search in genomic DNA. In *RECOMB* (April 2003), pp. 67–75.

[6] Burkhardt, S., and Kärkkäinen, J. Better filtering with gapped q-grams. In *Combinatorial Pattern Matching* (2001), pp. 73–85.

[7] Califano, A., and Rigoutsos, I. Flash: A fast look-up algorithm for string homology. In *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology* (July 1993), pp. 56–64.

[8] Crochemore, M., Landau, G., and Ziv-Ukelson, M. A sub-quadratic sequence alignment algorithm for unrestricted cost matrices. In *Proceedings of the Thirteen Annual ACM-SIAM Symposium on Discrete Algorithms* (2002), D. Eppstein, Ed., ACM-SIAM, pp. 679–688.

[9] Delcher, A., Kasif, S., Fleischmann, R., Peterson, J., White, O., and Steven, S. Alignement of whole genomes. *Nucleic Acids Reseach 27*, 11 (1999), 2369–2376.

[10] Florea, L., Hartzell, G., Zhang, Z., Rubin, G. M., and Miller, W. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Research 8* (1998), 967–974.

[11] Kent, W. J. BLAT–the BLAST-like alignment tool. *Genome Research 12* (2002), 656–664.

[12] Kolpakov, R., Bana, G., and Kucherov, G. mreps : efficient and flexible detection of tandem repeats in DNA sequences. *Nucleic Acid Research* (2003). Accepted for publication.

[13] Kurtz, S., and Schleiermacher, C. REPuter: Fast Computation of Maximal Repeats in complete genome. *Bioinformatics 15*, 5 (1999), 426–427.

[14] Lipman, D., and Pearson, W. Rapid and sensitive protein similarity searches. *Science 227* (1985), 1435–1441.

[15] Lipman, D., and Pearson, W. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA 85* (1988), 2444–2448.

[16] Ma, B., Tromp, J., and Li, M. PatternHunter: Faster and more sensitive homology search. *Bioinformatics 18*, 3 (2002), 440–445.

[17] Ning, Z., Cox, A. J., and Mullikin, J. C. SSAHA: A fast search method for large DNA databases. *Genome Research 11* (2001), 1725–1729.

[18] Schwartz, S., Kent, J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R., Haussler, D., and Miller, W. Human–mouse alignments with BLASTZ. *Genome Research 13* (2003), 103–107.

[19] Schwartz, S., Zhang, Z., Frazer, K., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. PipMaker – a web server for aligning two genomic DNA sequences. *Genome Research 10*, 4 (April 2000), 577–586.

[20] Smith, T., and Waterman, M. Identification of common molecular subsequences. *Journal of Molecular Biology 147*, 195–197 (Mar 1981).

[21] States, D., and Gish, W. Identification of protein coding regions by database similarity search. *Nature Genetics 3* (March 1993), 266–272.

[22] Ukkonen, E. On-line construction of suffix-trees. *Algorithmica 14* (1995), 249–260.

[23] Vincens, P., Buffat, L., André, C., Chevrolat, J.-P., Boisvieux, J.-F., and Hazout, S. A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics 14*, 8 (1998), 715–725.

[24] Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. A greedy algorithm for aligning DNA sequences. *Journal of Computational Biology 7*, 1/2 (2000), 203–214.

Contents

1 Introduction **3**

2 Algorithm Overview **4**

3 Linking Algorithm **5**

4 Extension algorithm **8**

5 Spaced seeds **12**

6 Results and Experiments **15**

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399