Identifying satellites and periodic repetitions in biological sequences

Marie-France Sagot [1,3] *

Eugene W. Myers [2]

[1] Service d'Informatique Scientifique
Institut Pasteur
28, rue du Dr. Roux
75724 Paris cedex 15
Tél.: (33) (1) 40 61 34 61
Fax : (33) (1) 40 61 30 80

[2] Department of Computer Science
University of Arizona
Tucson, AZ 85721-0099

[3] Institut Gaspard Monge
Université de Marne-la-Vallée
Cité Descartes - Champs-sur-Marne
77454 Marne-la-Vallée cedex 2

Abstract

We present in this paper an algorithm for identifying satellites in DNA sequences. Satellites (simple, micro, or mini) are repeats in number between 30 and as many as 1,000,000 whose lengths vary between 2 and hundreds of base pairs and that appear, with some mutations, in *tandem* along the sequence. We concentrate here on short to moderately long (up to 30-40 base pairs) approximate tandem repeats where copies may differ up to ϵ = 15-20% from a consensus model of the repeating unit (implying individual units may vary by 2ϵ from each other). The algorithm is composed of two parts. The first one consists of a filter that basically eliminates all regions whose probability of containing a satellite is less than one in 10^4 when $\epsilon = 10\%$. The second part realizes an exhaustive exploration of the space of all possible models for the repeating units present in the sequence. It therefore has the advantage over previous work of being able to report a consensus model, say m, of the repeated unit as well as the span of the satellite. The first phase was designed for efficiency and takes only $O(n)$ time where n is the length of the sequence. The second phase was designed for sensitivity and takes time $O(n \cdot \mathcal{N}(e,k))$ in the worst case where k is the length of the repeating unit m, $e = \lfloor \epsilon k \rfloor$ is the number of differences allowed between each repeat unit and the model m, and $\mathcal{N}(e,k)$ is the maximum number of words that are not more than e differences from another word of length k. That is, $\mathcal{N}(e,k)$ is the maximum size of an *e-neighborhood* of a string of length k. Experiments reveal the second phase to be considerably faster in practice than the worst-case complexity bound suggests. Finally, the present algorithm is easily adapted to finding tandem repeats in protein sequences, as well as extended to identifying mixed direct-inverse tandem repeats.

Keywords : *DNA satellites, tandem repeats, approximate match, consensus model, direct/inverse repeats.*

1 Introduction

We present an algorithm for identifying a series of *tandem repeats* in DNA sequences, that is a sequence of repeats that are adjacent in the sequence. Such tandemly repeated units are divided into three categories depending on the length of the repeated element, the span of the repeat region, and location within the chromosome [3]. Repeats occurring in or near the centromeres and telomeres are called simply *satellites*. Their span is large, up to a million bases, and the length of the repeated element varies greatly, anywhere from 5 to a few hundreds of base pairs. In the remaining, euchromatic region, of the chromosome the kinds of

*author to whom correspondence should be sent

tandem repeats found are classified as either *micro* or *mini satellites*, according to the length of the repeated element. Micro satellites are composed of short units, of 2 to 5 base pairs, in copy numbers in general around 100. Mini satellites on the other hand involve slightly longer repeats, typically around 15 base pairs, in clusters of variable sizes, comprising between 30 and 2000 elements. The functional role of satellites is not currently understood, but they tend to be highly polymorphic, and thus at a minimum are very useful as genetic markers. Searching for these repeats in new DNA sequence is standard practice amongst sequence analysts.

The few previous papers on this problem can be divided into three categories. The first concerns repeats that are exact (Karp [10], Crochemore [5], Clift [4], Milosavljevic [13] and Vérin [17]) or involve only two elements, that is, are of the form $\bar{u}\hat{u}$ where $\bar{u}$ and $\hat{u}$ are two words, either at some maximum edit distance from one another (Landau [11]), or, more generally, having a highest scoring alignment under a real-valued scoring system (Kannan and Myers [8]). Algorithms of the second group assume knowledge of the repeating unit, or assume their length is short enough that all words of that length can be generated and fitted to the sequence (Delgrange [6], Fischetti [7] and Rivals [15]). Finally, the third kind of approach has none of the previous limitations but resorts to heuristics in order to find the repeats (Benson [1] [2], Leung [9] [12] and Rivals [16]).

Our algorithm makes no assumptions about the repeat other than that its length be moderately short (up to 30-40 bases) and that the difference ratio ϵ between instances of the units and a consensus model of the repeat is not too high (up to 15-20%). It consists of two phases: a rapid, probabilistic filtering phase that eliminates 80% or more of the sequence from further consideration, followed by a model-driven backtrack search that is guaranteed to find in the remaining regions all satellites that verify certain properties.

The filtering phase is not exact in the sense that it cannot certify with absolute certainty that satellites are not present in the regions it eliminates. However, given a set of parameters (as, for instance, repeats length and number of repetitions), and assuming nucleotides occur with equal probability, we can and have estimated by simulation the chance of missing a satellite. Typically, this chance is very low, on the order of 1 in 10^4 when $\epsilon = 10\%$. On the other hand, real sequences tend to have significant bias in the frequency of bases over considerable stretches, e.g. some regions are 70-75% "GC-rich". These biases do not change the chance of false negatives, but can greatly increase the number of false positives that are reported, significantly reducing filtration efficiency. Even in such situations, the filter still eliminates as much as 80% of the sequence. The filter takes $O(nk)$ time where n is the length of the sequence and k is the maximum repeat unit length of the satellites sought.

The second phase of our algorithm effects an exhaustive search of the space of possible models for the repeating units present in the sequence. The essential feature is that this phase compares potential *satellite models* against the sequence to see if they could be the prefix of a consensus model of a satellite, and then exhaustively considers their extensions if so. Such a *valid* model prefix must approximately match a series of instances in the sequence within threshold $e = \lfloor \epsilon k \rfloor$ that are furthermore uniformly separated along the sequence. We call each matching instance a *wagon* and the sequence of wagons a *train*. A form of progressive dynamic programming between the prefix model and the sequence is used to keep track of the set of possible wagons and trains, and to effect the efficient extension of a promising prefix. To keep the search efficient it must be guided by a number of parameters, namely, one must specify a range $[min\_range, max\_range]$ for the length of the repeated unit and a minimum number, min_repeat, of units to be found in the satellite. Additional flexibility is introduced by allowing up to max_jump wagons of a train to be missing between two wagons that do match the model within the threshold e. This effectively allows us to find matches where some of the repeat instances are more than e-differences from the model. This second phase algorithm takes $O(n \cdot max\_jump \cdot max\_range^2 \cdot \mathcal{N}(e, max\_range))$ time and $O(n \cdot max\_range \cdot e)$ space, where $\mathcal{N}(e, k)$ is the maximum number of words at edit distance at most e from another word of length k.

The second phase of the algorithm does not require the filtering stage, so that one may, if desired, apply it directly to the sequence when exhaustiveness of the search is absolutely required. The algorithm becomes of course slower in this case, but where time is not a critical issue, the five to tenfold increase that is usually observed will not appear unreasonable. Forgoing the first phase enables us also to deal with repetitions that are not tandem, but simply periodic or approximately periodic. It further allows us to take into account the presence of inverted repeats flanked by direct ones [21]. Finally, the algorithm presented here may easily be extended to deal with tandem repeats in protein sequences as well.

This paper is organized as follows. We start by introducing some definitions, in particular concerning models, trains and wagons, and formally state the problem we propose to solve. We then describe the two phases of the algorithm in sections 3 (filter) and 4 (satellite identification). Satellites once found may be evaluated and this is shown in section 5. In section 6, we discuss some by-products of our algorithm and present an extension to deal with mixed/inverse tandem repeats [21]. Finally, in section 7 we give some performance results and an example of application.

2 Definitions and Statement of the Problem

In all that follows, we let s be the DNA target sequence in which satellites are being sought. We assume that s is of length n and over the DNA alphabet $\Sigma = \{A, C, G, T\}$. A *prefix model* of a satellite is a string m, not necessarily present itself in s, that approximately matches a train of wagons as alluded to in the introduction. Proceeding formally, let the e-neighborhood of m, $\mathcal{N}(e, m)$, be the set of all strings not more than e differences away from m, i.e., strings that can be transformed into m in at most e insertions, deletions, and substitutions. Recall that e is termed the *edit-distance* between the strings in question. Any substring of s that is also in $\mathcal{N}(e, m)$ is called a *wagon* of m. A *train* of m is a sequence of wagons $u_1, u_2, \ldots, u_p$ ordered by their starting positions in s and such that:

Property 1: $p \geq min\_repeat$ where min_repeat is a parameter chosen by the user that indicates the minimum number of elements a repeating region must contain.

Property 2: $left_{u_{i+1}} - left_{u_i} \in JUMP$ where $left_u$ is the position of the left-end of wagon u in s and:

$$JUMP = \{y : y \in \cup_{x \in [1, max\_jump]} \; x \times [min\_range, max\_range]\}$$

with the three parameters min_range, max_range and max_jump chosen by the user.

A prefix-model m is said to be *valid* if there is at least one train of m in the sequence s. Similarly, a train, when viewed simply as a sequence of substrings of s, is valid if it is the train for some model m. A prefix-model represents the invariant that must be true as we progressively search for our final goal, which is to arrive at a *consensus-model*. This is a prefix model that further satisfies the following:

Property 3: $left_{u_{i+1}} - right_{u_i} \in GAP$ where $right_u$ is the position of the right-end of wagon u, and $GAP = \{y : y \in \cup_{x \in [0, max\_jump-1]} \; x \times [min\_range, max\_range]\}$. Note that $0 \in GAP$. This guarantees that, when jumps are not authorized, the repeats found are effectively tandem.

Observe that the parameter max_jump allows us to deal with very badly conserved elements inside a satellite (by actually not counting them) while we require that the satellite be relatively well conserved overall. Since mutations affecting a unit concern indels as well as substitutions, it may be interesting to work with a variant of the above properties where $JUMP$ and GAP are defined as follows:

$$JUMP = \{y : \begin{array}{l} y \in [min\_range, max\_range] \; or \\ y \in \cup_{x \in [2, max\_jump]} \; x \times [min\_range - g, max\_range + g] \end{array} \}$$

$$GAP = \{y : \begin{array}{l} y \in [min\_range, max\_range] \; or \\ y \in \cup_{x \in [1, max\_jump-1]} \; x \times [min\_range - g, max\_range + g] \end{array} \}$$

where $g \geq e$ is a value chosen by the user. The idea is to allow the length of the badly conserved elements to vary more than is permitted for the detected wagons.

Finally, one can further extend the model m to be what was called a weighted combinatorial cover over Σ as in [19], effectively permitting one to find repeated patterns including the class of limited-regular expressions [22].

We conclude this section with the statement of our version of the satellite problem:

The Satellite Model Problem: Given a sequence s and parameters min_repeat, min_range, max_range, max_jump, and e (possibly also g), find all consensus models m that are valid for s, and for each such m, report a set of disjoint "fittest" trains realizing it.

3 Filtering

When an exhaustive search through progressively longer prefix models fails to end at a final consensus model, then we can be guaranteed that a repeat meeting the constraints of the problem parameters does not occur. While we show that such searches can be performed surprisingly efficiently, it is still desirable in most cases to first attempt to filter out all portions of s from further consideration with a certain empirical guarantee that little may be missed by doing so.

The basic idea behind the filtering step is that, if there is a satellite m^r, then in the dynamic programming matrix of s against itself, there should be a local alignment of length $p \cdot (r-i)$ centered on diagonal $i \cdot p$ for $i \in [1, r-1]$ and $p = |m|$. The similarity in these local alignments and the extent to which they lie on the given diagonal or at least near it, depends on the fidelity of the repeat units in the satellite. Generally, these off-diagonal alignments, corresponding to offset copies of the satellite, are strong enough to detect, provided the difference between the offset copies is at most 30-40%, which translates to a 15-20% difference between the repeat prototype and the instances in the satellite.

The specific heuristic we arrived at is to incrementally compute the local dynamic programming matrix of s versus itself in a scan of s, where after scanning s_i we have the matrix in the triangle $(i - band, i - band)$ to $(i - band, i)$ to (i, i). The parameter $band$ is chosen so that it is at least $\max\{3.5 \cdot max\_range, 50\}$, thus guaranteeing that at least three of the off-diagonal alignments of the satellite will be in the triangular region. Clearly, the total time taken is $O(n \cdot max\_range)$ as the algorithm computes an $O(max\_range)$ band about the diagonal.

For each potential period p of interest we compute the sum, $S(p)$, of the values in all the cells in diagonals $\{p \cdot i + \delta : i \in [1, \lfloor band/p \rfloor] \text{ and } \delta \in [-w(p), w(p)]\}$ where $w(p) = \max\{2, \epsilon \cdot p\}$. Effectively, we are taking the sum of the values in a $2 \cdot w(p)$ strip about each off-diagonal at period p, within the triangle. We keep this sum for each $p \in [min\_range, max\_range]$ at an additional factor of $O(w(max\_range) - w(min\_range))$ over the cost of dynamic programming in the band. What one expects is that in regions occupied by a satellite of period p, $S(p)$ is significantly larger than it is when the sequence is random, to the point where one can distinguish between the two.

Table 1 shows the results of a number of experiments. We ran 1000 trials on either a random sequence, or one containing a length 4, 8, or 16 repeat at various fidelities ϵ. For the local alignment objective, we used a scoring scheme that scored matches 1, and differences -2. In all cases, it is only till one gets to $\epsilon = 20\%$ that the distribution of random scores and those of the satellite filter scores begin to overlap, and only then by .2% of the 1000 trials with the indicated separating values. On several experiments with 10,000 trials, we were able to get only 1 overlap with ϵ at 10%, supporting our claim that the filter misses only 1 in 10^4 satellites at that fidelity.

4 Identifying the Satellites

4.1 Idea

As in previous papers where a simpler form of models was used [18] [19], satellite models are constructed by increasing lengths, that is, a valid prefix-model of length $k+1$ is obtained from its prefix of length k. In order to determine if a model is valid, we must have some representation of the train or wagons that make it so. There are two possibilities:

- we can keep track of each validating train and its associated wagons, or

- we can keep track of individual wagons, and, on the fly, determine if they can be combined into validating trains.

The first possibility is appealing because model extension is straightforward. We would just have to verify, for each wagon of each train, whether it can be extended in relation to the extended model, and then count how many wagons remain to see whether the train it belonged to remains a valid one. However, there are generally many overlapping trains involving many of the same wagons for a given model. Common wagons may therefore be present more than once in the list of occurrences of m if this is kept as a list of trains. So this approach entails redundancies that lead to an inefficient algorithm. We thus take the second approach, of keeping track of wagons and determining if they can be assembled into trains as needed.

The rules of prefix-model extension are given in Lemma 4.1 below. A wagon is identified by a triple (i, j, d) indicating that it is the substring $s_{i+1}s_{i+2}\dots s_j$ of s and is $d \leq e$ differences from its model. Thus i is the position of the left-end of the wagon, and j its right-end. Note carefully that in our convention, position i is *between* s_i and s_{i+1}.

> **Lemma 4.1**: The triple $(i, i+h, d)$ encodes a wagon of $m' = \alpha m$ with $\alpha \in \Sigma$ and $m \in \Sigma^k$ if and only if at least one of the following conditions is true:
>
> **(match & insert)**
> $(i+g, i+h, d-(g-1))$ is a wagon of m and $s_{i+1} = \alpha$.
> **(substitute)**
> $(i+1, i+h, d-1)$ is a wagon of m and $s_{i+1} \neq \alpha$.
> **(delete)**
> $(i, i+h, d-1)$ is a wagon of m.

For each prefix-model m, we keep a list of the wagons of m that are in at least one train validating m. We describe such wagons as being valid with respect to m. When we extend a model (to the left) to $m' = \alpha m$, we perform two tasks:

- First, determine which valid wagons of m can be extended as above to become wagons of m'.
- Second, of these newly determined wagons of m', keep only those that are valid with respect to m'. This requires effectively assemblying wagons into trains, something that is not needed in an approach that would keep track of trains directly.

Note that we need not actually enumerate the trains in the second step, we simply must determine if a wagon is part of one. This will allow us to perform an extension step in time linear with the length of the sequence.

As a final insight, consider the directed graph $G = (V, E)$ where V is the set of all valid wagons and there is an edge from wagon u to v if $left_v - left_u \in JUMP$. Then a wagon u is valid if it is part of a path of length *min_repeat* or more in G. Determining this is quite simple as the graph is clearly acyclic. In the computation that will follow, we will effectively compute the length of the longest path to u in $Lcnt_u$ and the length of the longest path from u in $Rcnt_u$. If $Lcnt_u + Rcnt_u > min\_repeat$ then u is valid.

4.2 Basic Procedure

We encode the collection of all wagons of m in a set, $L_m \subseteq \{0, 1 \dots, n\}$, and an $(n+1) \times (2e+1)$-element array D_m as follows:

1. $i \in L_m$ if and only if i is the left-end of at least one wagon valid with respect to m.
2. for each $i \in L_m$, the value $D_m[i, \delta]$ for $\delta \in [-e, e]$ is the edit distance of m from wagon $s_{i+1}s_{i+2}\dots s_{i+|m|+\delta}$.

Recall we adopted the convention that a wagon consisting of symbols $s_{i+1}s_{i+2}\ldots s_j$ has its left-end at index i and its right-end at index j, i.e. a position is *between* symbols, position i being between s_i and s_{i+1}. Intuitively, L_m gives the left-ends of all valid wagons which is all we need to verify Properties 1 and 2. D_m gives us the distances we need for extending models, together with the right-ends needed for verifying Property 3. Formally, $(i, i+|m|+\delta, d)$ is a valid wagon of m if and only if $i \in L_m$ and $d = D_m[i,\delta] \le e$.

The complete algorithm is given in Figure 1. When Extend(αm) is called, it is assumed that L_m is known along with the relevant D_m values. The routine computes these items for the extension αm and recursively for the extensions thereof. Lines 0-5 compute the set of left-ends of wagons for αm derivable from wagons of m that are valid. While Lemma 4.1 gives us a way to do so, we instead use dynamic programming to compute all extensions simultaneously. We thus prefer to think of the algorithm as adding the last row to the dynamic programming matrix of s versus αm. At the start L_m gives all the positions in row $|m|$ that have value e or less (and are valid) and D_m gives their values. From these, we compute the positions in row $|m|+1$ in the obvious sparse fashion to arrive at $L_{\alpha m}$ and the values $D_{\alpha m}$.

int $Lcnt[0..n]$, $Rcnt[0..n]$

procedure Extend(αm)
{ $L_{\alpha m} \leftarrow \emptyset$
1. **for** $i+1 \in L_m$ (in decreasing order) **do**
2. { **for** $\delta \in [-e, e]$ **do**
3. $$D_{\alpha m}[i,\delta] \leftarrow \min \left\{ \begin{array}{l} D_m[i+1,\delta] + (\textbf{if } s_i = \alpha \textbf{ then } \mathbf{0} \textbf{ else } \mathbf{1}), \\ \textbf{if } i \in L_m \textbf{ then } D_m[i,\delta+1]+1, \\ \textbf{if } i{+}1 \in L_{\alpha m} \textbf{ then } D_{\alpha m}[i+1,\delta-1]+1 \end{array} \right\}$$
4. **if** $min_\delta\{D_{\alpha m}[i,\delta]\} \le e$ **then**
5. $L_{\alpha m} \leftarrow L_{\alpha m} \cup \{i\}$
}

6. **for** $i \in L_{\alpha m}$ (in decreasing order) **do**
7. $Rcnt[i] \leftarrow \max_{k \in (i+\mathrm{JUMP}) \cap L_{\alpha m}}\{Rcnt[k]\} + 1$
8. **for** $i \in L_{\alpha m}$ (in increasing order) **do**
9. $Lcnt[i] \leftarrow \max_{k \in (i-\mathrm{JUMP}) \cap L_{\alpha m}}\{Lcnt[k]\} + 1$
10. **for** $i \in L_{\alpha m}$ **do**
11. **if** $Lcnt[i] + Rcnt[i] \le min\_repeat$ **then** $L_{\alpha m} \leftarrow L_{\alpha m} - \{i\}$

12. **if** $L_{\alpha m} \ne \emptyset$ **then**
13. { **if** $|\alpha m| \in [min\_range, max\_range]$ **then**
14. Record(αm) (see Section 5)
15. **if** $|\alpha m| < max\_range$ **then**
16. **for** $\beta \in \Sigma$ **do**
17. Extend($\beta\alpha m$)
}
}

Figure 1: Sketch of procedure for satellite model extension.

Once wagons have been extended when possible, we have to eliminate those that are no longer valid. This is performed by Lines 6 to 11. We compute, for each position $i \in L_{\alpha m}$, the maximum number of wagons in a train starting with a wagon whose left-end is at i in $Rcnt[i]$ (including itself), and the maximum number of wagons in a train ending with a wagon whose left-end is at i in $Lcnt[i]$. The necessary recurrences are given in Lines 7 and 9 of the algorithm where we recall that $JUMP = \{y : y \in \bigcup_{x\in[1,max\_jump]} x \times [min\_range, max\_range]\}$ and $i + JUMP$ denotes adding i to each element of $JUMP$. Observe that $Rcnt[i] + Lcnt[i] - 1$ is the length of the longest train containing a wagon whose left-end is at position i.

Clearly Lines 6-9 take $O(|L_{\alpha m}||JUMP|)$ time. However, when $L_{\alpha m}$ is a very large fraction of n, one can maintain an $Rcnt(Lcnt)$-prioritized queue of the positions in $(i + JUMP) \cap L_{\alpha m}$, to obtain an $O(n \cdot max\_jump \cdot \log |JUMP|)$ bound.

Finally in the remaining steps, Lines 12-17, the algorithm calls *Record* to record potential models and then recursively tries to extend the model if possible. The routine *Record* to be described in the next section, confirms that the model is a consensus model by verifying Property 3 and recording the intervals spanned by trains valid for the consensus model, if any.

4.3 Complexity

Proposition 4.3: The time complexity of the algorithm is bounded above by $O(n \cdot max\_range^2 \cdot max\_jump \cdot \mathcal{N}(e, max\_range))$. Its space requirement is at most $O(n \cdot max\_range \cdot e)$.

Proof While it is difficult to characterize the number of times the procedure *Extend* is called, it is still possible to arrive at an upper bound on the time taken by the algorithm of Figure 1.

We start with the observation that $O(|JUMP| + e)$ time is spent on a given left-end position for each prefix model matching the string beginning at that position. The factor e comes from Lines 2 and 3 while $|JUMP|$ is the number of back or forward elements that have to be examined in order to determine, for each wagon, the length of the longest train it may belong to (Lines 9 and 11). The number of prefix models that could match a given position with e or less errors is by definition $\Sigma_{k=1}^{max\_range} \mathcal{N}(e, k)$ where $\mathcal{N}(e, k)$ is the maximum number of words that are not more than e differences from another word of length k. That is, $\mathcal{N}(e, k)$ is the maximum size of an e-neighborhood of a string of length k. Thus the total time taken by the algorithm is bounded above by $O(n \cdot (|JUMP| + e) \cdot max\_range \cdot \mathcal{N}(e, max\_range)) = O(n \cdot max\_range^2 \cdot max\_jump \cdot \mathcal{N}(e, max\_range))$ as $e < max\_range$.

The space requirement is that of keeping all the information concerning at most *max_range* models at a time (a model m and all its prefixes). It is therefore on the order of at most $O(n \cdot max\_range \cdot e)$ as only $O(ne)$ storage is required to record the left-end positions and edit-distance at each possible right-end. $\Box$

Observe that, as we showed in earlier work, [20], $\mathcal{N}(e, k)$ is bounded from above by $k^e|\Sigma|^e$. Note also that the worst-case time analysis just given is very pessimistic. Essentially, the algorithm explores the trie of all words. As it gets deeper into the trie, it is very likely that no validating train will exist for the current model and one will backtrack on the trie. A probabilistic analysis is beyond the scope of the current paper, but the times reported in the empirical section certainly confirm this intuition.

5 Consensus Model Verification and Evaluation

5.1 Consensus Model Verification and Its Spans

Recall that our final goal is to arrive at consensus models, that is, at prefix models that further verify Property 3. The call to the routine *Record* in Step 14 of Figure 1 requires thus that, for each model m whose length is in the interval $[min\_range, max\_range]$, this property be checked.

In order to do this, we start with L_m and D_m, and note that if $i \in L_m$ is the left-end position of a valid wagon, then $Right(i) = \{i + |m| + \delta : D_m[i, \delta] \leq e\}$ is the set of right-end positions of valid wagons whose left-end is at i. Similarly $Left(j) = \{j - |m| - \delta : D_m[j - |m| - \delta, \delta] \leq e\}$ is the set of left-end positions of valid wagons whose right-end is at j. It follows that $R_m = \cup_{i\in L_m} Right(i)$ is the set of right-ends of valid wagons of m and it is easy to compute this set in $O(ne)$ time given L_m and D_m.

Consider now a directed bipartite graph $G_m = (L_m \cup R_m, E)$ whose vertices are the positions at which the left- and right-ends of valid wagons occur. Let there be a *wagon edge* $i \rightarrow j$ if and only if $j \in Right(i)$.

Further, let there be a *gap edge* $j \to k$ if and only if $k - j \in GAP$. Thus the edge sequence $i \to j \to k$ occurs in G_m if and only if there are valid wagons u and v such that $u = s_{i+1}s_{i+2}\cdots s_j$, $left_v = k$, and $left_v - right_u$ satisfies Property 3. It follows that a position/vertex which is on a path of length $2min\_repeat$ or more is part of a valid train satisfying all three properties, and is called a *final position*. Let G'_m be the graph induced by the set, F_m, of all final vertices. If G'_m is non-empty then m is a consensus model.

Each weakly-connected component of G'_m corresponds to a collection of overlapping trains. One should note that such a set of overlapping wagons/trains for a given model does indeed frequently occur. As an example if s = AGAAAAAAAGAATTTAGAAGAACAA, and the search parameters are $e = 1$, min_repeat = 3, min_range = 2, max_range = 4, and max_jump = 1, then G' for the model m = AGA consists of two components whose trains span the intervals $[1, 12]$ and $[16, 25]$. For instance, the left-end position sequences (1,4,7), (1,4,7,10), (4,7,10), (1,4,6,8,10), (1,3,6), (1,3,6,9), (3,6,9), (16,19,22), (16,18,20,22) among others, are all valid train-occurrences.

Let $[I, J]$ be the interval spanned by a weakly-connected component, $C \subseteq F_m$, where $I = min_{i\in C}\ i$ and $J = max_{j\in C}\ j$. This interval, called a *satellite span* of m, clearly contains every substring corresponding to a valid train within the component. The routine *Record* is responsible for determining all satellite spans of m as well as determining if m is a consensus model.

The graph G_m has at most $O(n)$ vertices and $O(n(e + |GAP|))$ edges. Thus we could explicitly build the graph G_m and then compute, with conventional linear graph algorithms, (1) the set of final positions F_m, (2) the subgraph G'_m it induces, and (3) the satellite spans for m, in $O(n(e + |GAP|))$ time and space. We can improve space to $O(n)$ by determining the edges of G_m on the fly, as follows. As for the basic procedure, we compute arrays *Lcnt*, *Lmrk*, *Rcnt*, and *Rmrk* in ordered sweeps of the merge of L_m and R_m. In this context we count *both* left and right ends, so that, for example, $Rcnt[i]$ for $i \in L_m$ will be the number of endpoints in the longest train (satisfying all 3 properties) starting at i, and $Rmrk[j]$ for $j \in R_m$ will be the number of endpoints in the longest train starting at j. After this computation in lines 1-6, it is clear that a left end $i \in L_m$ is final iff $Lcnt[i] + Rcnt[i] > 2min\_repeat$ and a right end $j \in R_m$ is final iff $Lmrk[i] + Rmrk[i] > 2min\_repeat$. In steps 7-12, with the aid of the recursive routines $RMark$ and $LMark$, we effect a marked traversal of G'_m that computes the satellite span $[I, J]$ of each connected component as it is traversed. After completing the traversal of a component, its span is stored with all other such spans (Step 12) as sketched in the next subsection.

5.2 Selecting Models to Report: Model Fitness

It often happens that the same, or approximately the same region corresponds to more than one consensus model, e.g. models ACA, ATA, and AAA are also valid for the two satellite spans in the example of the previous subsection. It is also often the case that the models are permuted versions of one another, e.g. models AGA, GAA and AAG. Furthermore, the models covering a given span can have different lengths, e.g. AGAA also covers the span $[1, 12]$ above. Finally, it may happen that one model is a *power* of the other: if, for instance, the minimum number of repeats in the example is lowered to 2, then both AGA and AGAAGA = AGA^2 are valid models, and we say that AGA is a *root* of AGAAGA. Frequently, models having overlapping satellite spans will differ in terms of a few wagons at the beginning or end of trains, and possibly of a few wagons in the middle when jumps are allowed.

In light of these observations, we have chosen to combine all overlapping spans into one large span and to associate with it all the models contributing satellite spans to it. This merging of the intervals takes place as the spans for individual models are found by *Record* so that at the completion of the algorithm of Figures 1 and 2, we have a list of disjoint intervals of s called satellite regions and for each region R, the set, $\mathcal{M}_R$, of models having a satellite span within R.

Given that $\mathcal{M}_R$ frequently contains more than a single model, we rank each $m \in \mathcal{M}_R$ according to the *fitness* of its match to R, which we define as the best scoring alignment between $(m)^\star$ and a substring of R under a user-specified scoring scheme δ. The efficient comparison of the regular expression $(m)^\star$ (matching zero or more repetitions of m) was introduced by Myers and Miller [14] for any regular expression, and a later reincarnation of their work for the case of this special pattern type became popularly known as *wrap around*

set of $[0..n]$ R
int I, J, $Lmrk[0..n]$, $Rmrk[0..n]$

procedure RMark(j)
{ "Mark j"
  if $j > J$ **then** $J \leftarrow j$
  for $i \in (Left(j) \cup (j + GAP) \cap L_m$ **do**
    if "i is unmarked" and $Lcnt[i] + Rcnt[i] > 2min\_repeat$ **then**
      LMark(i)
}

procedure LMark(i) "Similar to $Rmark$ above."

procedure Record(m)
{ $R \leftarrow \cup_{i \in L_m} Right(i)$
  for $i \in L_m \cup R$ (in decreasing order) **do**
    if $i \in R$ **then**
      $Rmrk[i] \leftarrow \max_{k \in (i+\mathrm{GAP}) \cap L_m}\{Rcnt[k]\} + 1$
    if $i \in L_m$ **then**
      $Rcnt[i] \leftarrow \max_{k \in Right(i)}\{Rmrk[k]\} + 1$
  "Similarly compute $Lcnt[i]$ for all $i \in L_m$ and $Lmrk[i]$ for all $i \in R$."
  "All positions in R and L_m are unmarked."
  for $j \in R$ **do**
    if "j is unmarked" and $Lmrk[j] + Rmrk[j] > 2min\_repeat$ **then**
    { $I \leftarrow J \leftarrow j$
      $RMark(j)$
      "Merge satellite span (I, J, m) with global region-and-models list."
    }
}

Figure 2: Procedure to determine if m is a consensus model and calculate its satellite spans.

dynamic programming [11]. A ranking of models according to such a fitness score for the region relegates secondary model matches, due to a fidelity mismatch between ϵ and that of the satellite, to the back of the list. Moreover, we further avoid reporting permutations or powers or roots of one another, by always picking the fitter of any two such models and removing the other. The complexity of this post-processing phase is dominated by the search for consensus models.

6 By-Products and Extension

6.1 By-Products

One of the by-products of the approach we introduced here for finding satellites is that our algorithm is also capable of identifying periodic, or approximately periodic repeats that may be non-contiguous. These correspond to prefix models, i.e. to models that verify Properties 1 and 2 but not necessarily Property 3.

Of course, wrap-around dynamic programming as given in subsection 5.2 cannot then be used to examine the fitness of the trains reported. One possibility would be to select the train for which the sum of the scores of the wagons is optimal and report this as the fitness of the model. An alternative to this could be to measure a model m fitness by comparing the regular expression $(m\Sigma^{p-|m|})^\star$ against the satellite span where p denotes the period of the repeat.

As suggested in the definitions, we can also extend the model m to permit finding repeated patterns that include the class of limited-regular expressions [22]. This allows for an easy adaptation of the algorithm that is particularly interesting for dealing with proteins. The search for tandem (or periodic) repeats in such molecules can then be performed in exactly the same way. An example of a simple application of this is given in section 7.2.

6.2 Tandem Direct and Inverted Repeats

A more substantial modification of the algorithm permits us to treat the presence of inverted repeats amongst the direct ones when these occur in tandem. The exercise of doing so entails distinguishing between direct and inverted wagons, extending Properties 2 and 3 to accommodate the inverted wagons, and then modifying the basic algorithm of Figures 1 and 2 accordingly.

The extension of Properties 2 and 3 gives raise to the following new versions:

(Property 2)': $end_{u_{i+1}} - end_{u_i} \in \begin{cases} JUMP & \text{if both } u_i \text{ and } u_{i+1} \text{ are direct} \\ & \text{or both are inverted} \\ JUMP\_INV & \text{otherwise} \end{cases}$

where $end_u = \begin{cases} left_u & \text{if } u \text{ is direct} \\ right_u & \text{otherwise} \end{cases}$ and:

$JUMP\_INV = \{y : y \in \bigcup_{x\in[0,max\_jump-1]} [x \times min\_range, (x+2) \times max\_range]\}$.

(Property 3)': $start_{u_{i+1}} - end_{u_i} \in GAP$

where $start_u = \begin{cases} right_u & \text{if } u \text{ is direct} \\ left_u & \text{otherwise} \end{cases}$ and $end_u = \begin{cases} left_u & \text{if } u \text{ is direct} \\ right_u & \text{otherwise.} \end{cases}$

A wagon is now identified by a quadruple (i, j, d, f) where the additional variable f is a flag indicating whether we are dealing with a direct or an inverted occurrence. Lemma 6.2 thus becomes:

Lemma 6.2: The quadruple $(i, i + h, d, f)$ encodes a wagon of $m' = \alpha m$ with $\alpha \in \Sigma$ and $m \in \Sigma^k$ if and only if,

(f = direct) at least one of the conditions of lemma 4.1 are verified;

(f = indirect) at least one of the following conditions is true:

(match & insert)
$(i, i + h - g, d - (g - 1), f)$ is a wagon of m and $s_{i+h} = \alpha$.
(substitute)
$(i, i + h - 1, d - 1, f)$ is a wagon of m and $s_{i+h} \neq \alpha$.
(delete)
$(i, i + h, d - 1, f)$ is a wagon of m.

The only thing that needs changing in the algorithms of Figures 1 and 2 are the lines concerning train and contiguity checking. These are lines 6 to 9 in Figure 1 and lines 3 to 5 (and corresponding computations for $Lcnt[i]$ mentioned in Line 6) in Figure 2. We sketch in Figure 3 how lines 6 and 7 in Figure 1 should now read. Modifying the remaining parts is done in a similar way.

Finally, for this version of the problem, fitness would be measured by comparing the regular expression $(m|\overline{m})^\star$ against the satellite span where $\overline{m}$ denotes the inverse of model m.

6. **for** $(i, f_i) \in L_{\alpha m}$ (in decreasing order) **do**
7. $\quad Rcnt[i] \leftarrow \max_{(k,f_k)\in \begin{cases} (i+\text{JUMP}) \cap L_{\alpha m} & \text{if} f_i = f_k \\ (i+\text{JUMP\_INV}) \cap L_{\alpha m} & \text{otherwise} \end{cases}} \{Rcnt[k]\} + 1$

Figure 3: Train verification in the case of direct/inverse tandem repeats.

7 Applications

7.1 Some Empirical Performance Results

We chose to present an empirical evaluation of the performance of our algorithm by searching all the chromosomes of yeast Saccharomyces cerivisiae for satellites. We used for this a fairly spread out variety of parameter sets that we list in Table 2. The sequences were retrieved from the following WEB address: `ftp://ftp.mips.embnet.org/pub/yeast/`. In Table 3, we give the execution times obtained on a Dec Alpha 4000 5/466 for each run of our algorithm combined with the filter prepass. A longer execution than the average appearing in the table indicates in general that a satellite verifying the input parameters has been found and the algorithm needs then more time to finely build the corresponding consensus model. This is typically the case of chromosome IX as we show in next subsection. Conversely, a shorter time than the average usually means there is no satellite in the sequence that verifies the parameters even approximately and the search thus ends earlier.

7.2 A More Fully Developed Example

We now develop in fuller detail an example of application and use the results obtained to discuss some points of the algorithm.

The example is drawn from chromosome IX of yeast Saccharomyces cerivisiae. Figure 4 shows a tandem repeat starting at position 391 131 (in the sequence as recovered from the WEB address given above) of the chromosome. This repeat is composed of 41 full units, 16 of which present a deletion of 9 bp. as against the other elements. Apart from this, the repeat is well conserved overall (on average one mutated base per element), except for the first six units and for the last one. The repeat is located inside a coding region (in the other strand) and Figure 5 shows the DNA repeat (inversed and complemented) together with its corresponding translation. The translated repeat is part of a glucoamylase s1/s2 precursor protein (SwissProt identification: AMYH_YEAST).

Table 5 gives a summary of the consensus for the DNA repeat (as determined by the majority rule) and of the fittest satellite models found by us concerning this repeat under a scoring scheme that is as follows: matches, +1, mismatches, −1, and indels, −2 (it is thus more permissive of mismatches than indels). The other parameters of the algorithm were set as indicated in Table 4.

The results in Table 5 follow the order in which we obtained them as we progressively refined the parameters to get to the final, possibly correct, model for the repeat. Observe that, although the intervals of lengths we worked with in our first runs of the algorithm were too short to capture the consensus model for the satellite, we were nevertheless able, thanks to the fact that bad wagons were allowed to be jumped, to notice something of interest was present in the region. We managed even to get the satellite prefix model and span approximately right.

That we were getting various slightly different models with close enough scores may, on the other hand, have hinted to the fact that we were not accurately constraining our search, in particular concerning the period of repeat. However, it will often be the case that there will be more than one model for a satellite region not only because we are setting the parameters wrong for the identification of a given tandem repeat, but also because it is not clear what template or prototype sequence the repeated units came from. This could suggest working with an edit distance, even a weighted one, as a similarity measure between repeats

```
GTTGCTAGAGGAAGATGGGGTTGGTACTGGTGCTACAGAGCTTTC
AGTGGTGGAGCTGGAT         ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTTGATGGGGTTGGTACTGGAACAGAAGAGCTTTC
AGTGCTAGAGCTGAATGGGGTTGAAGATGGAGCGGAGGAAGTGAT
GTTGCTAGAGGAAGATGGGGTTGGTACTGGTGCTACAGAGCTTTC
AGTAGTAGAGCTTGATGGGGTTGGTACTGGAGCAGAAGAGCTTTC
GGTAGTAGAGCTGGATGGAGTTGGCACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGGGTTGGTACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTTGATGGGGTTGGAGCTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTTGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTTGATGGAGTTGGCACTGGAGCAGAAGAGCTTTC
AGTGGTGGAGCTGG      TT   ACTGGAGTAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTGGTGGAGCTTGATGGGGTTGGAGCTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTTGATGGGGTTGGTACTGGAGCAGAAGAGCTTTC
AGTGGTAGAGCTGG      TT   ACTGGAGCAGAAGAGCTTTC
AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAACTTTC
AGTAGTAGAGCTTGATGGGGTTGGTACTGGAGTAGTAGTCTTCTT
```

Figure 4: An example of a tandem repeat in chromosome IX of yeast Saccharomyces cerivisiae, starting at position 391 131 (in the sequence as recovered from the WEB address given in the text).

```
AAGAAGACTACTACTCCAGTACCAACCCCATCAAGCTCTACTACT         KKTTTPVPTPSSSTT
GAAAGTTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACCCCATCAAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGCTCCAACCCCATCAAGCTCCACCACT         ESSSAPAPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTACTCCAGT   AA      CCAGCTCCACCACT         ESSSTPV T SS TT
GAAAGCTCTTCTGCTCCAGTGCCAACTCCATCAAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCAAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGCTCCAACCCCATCAAGCTCTACTACT         ESSSAPAPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTACCAACCCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT   AA      CCAGCTCTACCACT         ESSSAPV T SS TT
GAAAGCTCTTCTGCTCCAGTACCAACTCCATCCAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTGCCAACTCCATCCAGCTCTACTACC         ESSSAPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGTACCAACCCCATCAAGCTCTACTACT         ESSSAPVPTPSSSTT
GAAAGCTCTGTAGCACCAGTACCAACCCCATCTTCCTCTAGCAAC         ESSVAPVPTPSSSSN
ATCACTTCCTCCGCTCCATCTTCAACCCCATTCAGCTCTAGCACT         ITSSAPSSTPFSSST
GAAAGCTCTTCTGTTCCAGTACCAACCCCATCAAGCTCTACTACT         ESSSVPVPTPSSSTT
GAAAGCTCTTCTGCTCCAGT         ATCCAGCTCCACCACT         ESSSAPV   SSSTT
GAAAGCTCTGTAGCACCAGTACCAACCCCATCTTCCTCTAGCAAC         ESSVAPVPTPSSSSN
```

\

Figure 5: The same tandem repeat as in Figure 4, inversed and complemented (left) together with its translation (right). The translated sequence is part of a glucoamylase s1/s2 precursor protein.

may not be the best way of capturing the whole flavor of such repeats.

Locating the repeat after translation is, in this case, a slightly easier affair as some of the substitutions at the DNA level occur at silent positions. Table 6 gives a summary of the consensus for the protein repeat (again, as determined by the majority rule) and of the fittest satellite model found by us concerning this repeat under the same scoring scheme as above (other schemes, in particular using substitution weight matrices, may also be employed). The parameters used for the algorithm in this case were set as follows: $e = 1$ (subs. only), $g = 1$, $min\_repeat = 15$, $min\_range = 13$, $max\_range = 18$, $max\_jump = 4$.

8 Future Work

We have presented an algorithm for identifying tandem repeats that is both efficient and sensitive. It requires only approximate knowledge of the length of the repeating unit and no knowledge at all of the repeat itself. A model for such a repeat, that may itself not occur in the sequence, is constructed simultaneously with the search. The approach introduced here may also handle periodic repetitions that are not contiguous and/or involve inversions.

The current algorithm is most appropriate for identifying short or moderately long repetitions and is not meant to treat the problem of very long (composed of hundreds of bases) repeats such as results from duplication, transposition or inversion events. Other methods have to be elaborated in this case, and this is one of the areas of research we wish to explore in future.

A subtle problem not considered in this paper either concerns the ability to more finely analyze the mutational structure of a repeat. Answers to questions such as: is there a regularity in the distribution of mutated positions of a tandem array, or in the nature of the mutations observed, may prove interesting biologically. These are challenging problems, as it is often difficult even to decide which model among those found best represents the possible originator of a repeat. As suggested, working with an edit distance may not be the best way to get at this originator and more phylogenetically appropriate similarity metrics may be required. These issues are biologically important but we do not know yet how to treat them algorithmically.

Finally, we have as yet not attempted to evaluate the statistical pertinence of the models found. This is a significant aspect of the *analysis* of the results of our algorithm and will be undertaken in a subsequent work.

Acknowledgments

Both authors would like to thank Alain Viari, Maxime Crochemore, Laurent Bloch and Antoine Danchin from, respectively, the Atelier de BioInformatique of the University of Paris VI, the Institut Gaspard Monge from the University of Marne-la-Vallée, and both the Service d'Informatique Scientifique and the Laboratoire de Régulation de l'Expression Génétique of the Institut Pasteur in Paris for their warm hospitality during Gene's stay in Paris in May 1997 while this work was being developed, as well as the Computer Science Department of the University of Arizona for its kind invitation of Marie-France in Feb/March 1998 while the journal version of this paper was being drafted.

Thanks is also due to Laurent Mouchard for a very careful reading of this paper, Guy-Franck Richard for having provided some very useful bibliographic indications on satellites and Frédérique Galisson from the Service d'Informatique Scientifique for having pointed us out to Guy-Franck Richard.

Finally, the first author would like to dedicate this paper to Alain Viari. The discussions she had with him in the earlier stages of the work were, as always, inspiring and extremely fruitful.

References

[1] G. Benson. An algorithm for finding tandem repeats of unspecified pattern size. In M. Waterman S. Istrail, P. Pevzner, editor, *RECOMB 98*, pages 20–29. ACM Press, 1998.

[2] G. Benson and M. Waterman. A method for fast database search for all k-nucleotide repeats. *Nucleic Acids Research*, 22:4828–4836, 1994.

[3] B. Charlesworth, P. Sniegowski, and W. Stephan. The evolutionary dynamics of repetitive DNA in eukaryotes. *Nature*, 371:215–220, 1994.

[4] B. Clift, D. Haussler, R. McConnell, T. D. Schneider, and G. D. Stormo. Sequence landscapes. *Nucleic Acids Res.*, 14:141–158, 1986.

[5] M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Proc. Letters*, 12:244–250, 1981.

[6] O. Delgrange. *Un algorithme rapide pour une compression modulaire optimale. Application à l'analyse de séquences génétiques*. PhD thesis, 1997. Thèse de doctorat - Université de Lille I.

[7] V. Fischetti, G. Landau, J. Schmidt, and P. Sellers. Identifying periodic occurrences of a template with applications to protein structure. In Z. Galil A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 111–120. Springer-Verlag, 1992.

[8] S. K. Kannan and E. W. Myers. An algorithm for locating non-overlapping regions of maximum alignment score. In Z. Galil A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, page 7486. Springer-Verlag, 1993.

[9] S. Karlin, M. Morris, G. Ghandour, and M.-Y. Leung. Efficient algorithms for molecular sequence analysis. *Proc. Natl. Acad. Sci. USA*, 85:841–845, 1988.

[10] R. M. Karp, R. E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *Proc. 4th Annu. ACM Symp. Theory of Computing*, pages 125–136, 1972.

[11] G. Landau and J. Schmidt. An algorithm for approximate tandem repeats. In Z. Galil A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 120–133. Springer-Verlag, 1993.

[12] M.-Y. Leung, B. E. Blaisdell, C. Burge, and S. Karlin. An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *J. Mol. Biol.*, 221:1367–1378, 1991.

[13] A. Milosavljevic and J. Jurka. Discovering simple DNA sequences by the algorithmic significance method. *Comput. Appl. Biosci.*, 9:407–411, 1993.

[14] E. W. Myers and W. Miller. Approximate matching of regular expressions. *Bull. Math. Biol.*, 51:5–37, 1989.

[15] E. Rivals and O. Delgrange. A first step toward chromosome analysis by compression algorithms. In N. G. Bourbakis, editor, *First International IEEE Symposium on Intelligence in Neural and Biological Systems*, pages 233–239. IEEE Computer Society Press, 1995.

[16] E. Rivals, O. Delgrange, J.-P. Delahaye, M. Dauchet, M.-O. Delorme, A. Hénaut, and E. Ollivier. Detection of significant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences. *Comput. Appl. Biosci.*, 13:131–136, 1997.

[17] R.Verin and M. Crochemore. Direct construction of compact directed acyclic word graphs. In A. Apostolico and J. Hein, editors, *Combinatorial Pattern Matching*, volume 1264 of *Lecture Notes in Computer Science*, pages 116–129. Springer-Verlag, 1997.

[18] M.-F. Sagot, V. Escalier, A. Viari, and H. Soldano. Searching for repeated words in a text allowing for mismatches and gaps. In R. Baeza-Yates and U. Manber, editors, *Second South American Workshop on String Processing*, pages 87–100, Viñas del Mar, Chili, 1995. University of Chili.

[19] M.-F. Sagot and A. Viari. A double combinatorial approach to discovering patterns in biological sequences. In D. Hirschberg and G. Myers, editors, *Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 186–208. Springer-Verlag, 1996.

[20] M.-F. Sagot, A. Viari, and H. Soldano. Multiple comparison: a peptide matching approach. *Theoret. Comput. Sci.*, 180:115–137, 1997. presented at *Combinatorial Pattern Matching 1995*.

[21] R. D. Wells and R. R. Sinden. Defined ordered sequence DNA, DNA structure and DNA-directed mutation. In K. E. Davies and S. T. Warren, editors, *Genome Analysis*, volume 7 of *Genome Rearrangement and Stability*, pages 107–138. Cold Spring Harbor Laboratory Press, 1993.

[22] S. Wu, U. Manber, and E. Myers. A sub-quadratic algorithm for approximate limited expression matching. *Algorithmica*, 15:50–67, 1996.

Repeat Period (p)	Random Maximum	$\epsilon = 10\%$ Min(Overlap)	$\epsilon = 15\%$ Min(Overlap)	$\epsilon = 20\%$ Min(Overlap)	Separating Value
4	1803	8947(0%)	3797(0%)	1741(.2%)	1800
8	1422	7947(0%)	2879(0%)	1311(.2%)	1350
16	1023	5777(0%)	2354(0%)	467(.2%)	850

Table 1: Filter discrimination between random and satellite sequence.

Parameter Set	e	min_repeat	min_range	max_range	max_jump
1	1	50	2	10	2
2	1	50	2	10	3
3	2	50	10	20	2
4	2	50	10	20	3
5	2	50	20	30	2
6	2	50	20	30	3
7	3	50	30	40	2
8	3	50	30	40	3
9	3	50	40	50	2
10	3	50	40	50	3

Table 2: Satellite parameter sets for the performance results given in Table 3.

Chromosome	Length (in bp.)	Parameter set									
		1	2	3	4	5	6	7	8	9	10
1	233096	23	27	45	56	3	4	4	5	4	5
2	823309	68	82	131	165	57	80	881	1093	298	388
3	321665	34	41	77	101	29	40	7	7	6	7
4	1551088	126	152	212	277	71	86	21	25	22	28
5	584090	53	65	114	147	47	58	9	11	9	11
6	273534	29	34	51	64	4	5	4	5	5	6
7	1104589	95	116	156	196	27	34	16	19	17	21
8	569680	48	61	93	114	20	25	8	9	9	10
9	445393	41	50	187	235	127	172	1904	2554	1291	1574
10	754768	55	67	113	142	91	129	1079	1307	11	12
11	674796	59	71	91	109	17	21	10	12	11	12
12	1091661	92	113	222	282	78	95	15	18	17	20
13	935996	77	94	179	242	104	133	448	539	15	17
14	794145	66	82	101	129	33	41	11	13	13	14
15	1104935	91	113	209	264	59	71	15	18	18	20
16	959922	79	96	157	200	46	56	13	16	14	17

Table 3: Execution times (in seconds) of the algorithm on a Dec Alpha 4000 5/466 for the parameter sets given in Table 2.

Parameter Set	e	g	min_repeat	min_range	max_range	max_jump
1	1	0	10	20	30	3
2	1	4	20	20	30	3
3	2(subs. only)	4	15	30	40	4
4	2(subs. only)	5	15	40	50	4

Table 4: Parameter sets for the results shown in Table 5 concerning the 45 bp satellite in Chromosome IX of Yeast.

	start	end	score	consensus or model	time
consensus	391 131	392 832		AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC	
model for param. 1	391 337	392 718	606	AGTAGTAGAGCAGAAGAGCTTTC	66 s.
	391 349	392 718	582	AGTAGTAGAGCTAGAAGAGCTTTC	
	391 349	392 717	573	AGTAGTGAGCAGAAGAGCTTTC	
model for param. 2	391 337	392 717	606	AGTAGTAGAGCAGAAGAGCTTTC	90 s.
	391 349	392 718	594	TGGTACTGGACAGAAGAGCT	
	391 349	392 717	586	GTACTGGAGCAGAAGAGCTTTC	
model for param. 3	391 379	392 675	871	AGTAGTAGAGCTGGTGGGTACTGGAGCAGAAGAGCTTTC	310 s.
	391 376	392 720	864	AGTAGTAGAGCTTGGTTACTGGAGCAGAAGAGCTTTC	
	391 378	392 675	858	AGTGGTAGAGCTGGTGGTACTGGAGCAGAAGAGCTTTC	
	391 338	392 718	855	AGTAGTAGAGCTGGTTACTGGAGCAGAAGAGCTTTC	
model for param. 4	391 338	392 741	1060	AGTAGTAGAGCTGGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC	970 s.
	391 350	392 741	1045	AGTAGTAGAGCTGATGGAGTTGGTACTGGAGCAGAAGAGCTTTC	

Table 5: Consensus obtained by majority rule for the 45 bp satellite in Chromosome IX of Yeast and satellite consensus models obtained by our algorithm for the various parameters indicated in Table 4.

	start	end	score	consensus or model	time
consensus	308	873		ESSSAPVPTPSSST	
model	313	872	436	ESSSAPVPTPSSST	107 s.

Table 6: Consensus obtained by majority rule for the 15 bp satellite in AMYH_YEAST and satellite consensus model obtained by our algorithm for the parameters as indicated in text.