Identifying satellites in nucleic acid sequences

Marie-France Sagot [1,3]   Eugene W. Myers [2]

[1] Service d'Informatique Scientifique
Institut Pasteur
28, rue du Dr. Roux - Paris

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

[3] Institut Gaspard Monge
Université de Marne la Vallée
2, rue de la Butte Verte - Noisy le Grand

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. Thus it 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 $O(n \cdot \mathcal{N}(e,k))$ 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.

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

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 [2]. 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 100 base pairs. In the remaining, euchromatic region, of the chromosome the kinds of 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 typically around 100. Mini satellites on the other hand involve slightly longer repeats, 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 [7] and Milosavljevic [10]) 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 [8]), or, more generally, having a highest scoring alignment under a real-valued scoring system (Kannan and Myers [5]). 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 [3], Fischetti [4] and Rivals [12]). Finally, the third kind of approach has none of the previous limitations but resorts to heuristics in order to find the repeats (Benson [1], Leung [6] [9] and Rivals [13]).

Our algorithm makes no assumptions about the repeat other than than 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 all satellites in the remaining regions by exhaustion.

The filtering phase is not exact in the sense that it cannot guarantee 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. 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 within 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. But 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. 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, and even allows us to take into account the presence of inverted repeats flanked by direct ones[17].

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 set of wagons $u_1, u_2, \ldots u_p$ 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 $JUMP = \{xy : x \in [1, max\_jump]$ and $y \in [min\_range, max\_range]\}$ 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 a consensus model of a satellite. Our final goal is to arrive at a *consensus-model* which 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 $GAP = \{xy : x \in [0, max\_jump-1]$ and $y \in [min\_range, max\_range]\}$. Note that $0 \in GAP$.

Observe that, as a by product of our approach, we will find all valid prefix-models and these represent approximately or exactly periodic, but non-contiguous repeats in s. If desired, one could easily develop a criterion for deciding which such prefix-models are significant and worthy of reporting. Secondly, note 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. Finally, one can further extend the model m to be what was called a weighted combinatorial cover over Σ as in [15], effectively permitting one to find repeated patterns including the class of limited-regular expressions [18].

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, 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 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 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 where $p = |m|$, 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 $i \in [1, r-1]$. 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 computation in total 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 $\{p \cdot i + \delta : i \in [1, \lfloor band/p \rfloor]$ 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 [14] [15], 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. However, there are generally many overlapping trains involving many of the same wagons for a given model. 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 $s_{i+1}s_{i+2}\ldots s_j$ of s and is $d \le e$ 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 $s_{i+g} = \alpha$.
> **(substitute)**
> $(i+1, i+h, d-1)$ is a wagon of m and $s_i \ne \alpha$.
> **(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 keeps 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 linear time.

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, $L_m \subseteq \{0, 1 \ldots, n\}$, 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 least edit distance of m from wagon $s_{i+1}s_{i+2}\ldots s_{i+|m|+\delta}$.

We adopt the convention that a wagon consisting of symbols $s_i s_{i+1} \ldots s_j$ has its left-end at index $i-1$ 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 what we need to verify Properties 1 and 2. D_m gives us the distances we need for extending models and 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

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

procedure Extend(αm)
{ $\quad L_{\alpha m} \leftarrow \emptyset$
1. **for** $i+1 \in L_m$ (in decreasing order) **do**
2. $\quad$ { **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. $\qquad$ **if** $min_\delta\{D_{\alpha m}[i,\delta]\} \leq e$ **then**
5. $\qquad\quad L_{\alpha m} \leftarrow L_{\alpha m} \cup \{i\}$
$\quad$ }

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

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

Figure 1: Sketch of procedure for satellite model extension.

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 $D_{\alpha m}$.

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 $JUMP = \{xy : x \in [1, max\_jump]$ and $y \in [min\_range, max\_range]\}$ and $i + JUMP$ denotes adding i to each element of $JUMP$. Observe that $Rcnt[i] + Lcnt[i]$ 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 Sketch of Complexity

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 by starting 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 number of such prefixes that could match the given position with e or less errors is by definition $\Sigma_{k=1}^{max\_range} \mathcal{N}(e,k) = O(max\_range \cdot \mathcal{N}(e, max\_range))$. 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 $e < max\_range$. We showed in earlier work, [16], that $\mathcal{N}(e,k)$ is bounded from above by $k^e|\Sigma|^e$.

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 least edit-distance at each possible right-end.

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. A probabilistic analysis is beyond the scope of the current paper, but the times reported in the empirical section certainly confirm this intuition.

5 Evaluation

5.1 Consensus Model Verification and Its Spans

The call to the routine *Record* in Step 14 of Figure 1 requires that we verify Property 3 for a model m whose length is in the interval $[min\_range, max\_range]$ and which satisfies Properties 1 and 2. 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 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* $i \to j$ 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. For example if s = AGAAAAAAA-GAATTTAGAAGAACAA, 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 example, the left-end position sequences (0,3,6), (0,3,6,9), (3,6,9), (0,3,5,7,9), (0,2,5), (0,2,5,8), (2,5,8), (15,18,21), (15,17,19,21) 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 if m is a consensus model and if so, then determining all satellite spans of m. 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 intervals of the example of the previous subsection. Furthermore, it is also often the case that the models are permuted versions of one another, e.g. models AGA, GAA and AAG. The models covering a given span can also have different lengths, e.g. AGAA also covers the span $[1, 12]$ above. It can also happen that one model is a *power* of the other. As an example, if the minimum number of repeats 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 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 [11] 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 dynamic programming* [8]. 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 is dominated by the search for consensus models.

6 Some Applications

We evaluated our algorithm by searching all the chromosomes of yeast Saccharomyces cerivisiae for satellites. The sequences were retrieved from the following WEB address: `ftp://ftp.mips.embnet.org/pub/yeast/`. Various parameter sets were tried and these are listed in Table 2. 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. Finally, Table 4 presents a summary of the satellite models found (all parameter sets considered together) that have a score above 100 under either of two different scoring schemes for the wrap-around dynamic programming fitness evaluation. *Score 1* is that used by Benson [1] and scores matches, +2, and differences, −6. *Score 2*, of our own design, scores matches, +1, mismatches, −1, and indels, −2. It is thus more permissive of mismatches than indels.

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

procedure RMark(j)
{   "Mark j"
    if j > J then J ← j
    for i ∈ (Left(j) ∪ (j + GAP) ∩ 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 ← ∪_{i∈L_m} Right(i)

    for i ∈ L_m ∪ R (in decreasing order) do
        if i ∈ R then
            Rmrk[i] ← max_{k∈(i+GAP)∩L_m} {Rcnt[k]} + 1
        if i ∈ L_m then
            Rcnt[i] ← max_{k∈Right(i)} {Rmrk[k]} + 1

    "Similarly compute Lcnt[i] for all i ∈ L_m and Lmrk[i] for all i ∈ R."

    "All positions in R and L_m are unmarked."
    for j ∈ R do
        if "j is unmarked" and Lmrk[j] + Rmrk[j] > 2min_repeat then
         {  I ← J ← 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.

Note that even after removing powers and selecting the best model of a given length, there can still be more than one model for a satellite region. It is interesting that in these cases it is not clear what template or prototype sequence the repeated units came from.

We have as yet not explored the results of the table for biological significance and we have 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.

7 Extensions

As mentioned in the introduction, if we forgo the use of the first phase filter, we can use the second phase to identify repeats that are not tandem yet occur with, approximately or exactly, the same period. Of course, wrap-around dynamic programming cannot be used to examine the fitness of the trains reported. A simple alternative would be to select the train for which the sum of the scores of the wagons is optimal.

Although not presently implemented, we may also treat the presence of inverted repeats among the direct ones when these occur in tandem. The exercise of doing so entails distinguishing between direct and inverted wagons, and extending properties 2 and 3 to accommodate the inverted wagons, and then modifying the basic algorithm of Figures 1 and 2 accordingly. For this version of the problem, fitness would be measured by comparing the regular expression $(m|\overline{m})^*$ against the satellite span where $\overline{m}$ denotes the inverse of model m.

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.

A subtle problem not considered here 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 we have seen here that it is often difficult even to decide which model among those found best represents the possible originator of a repeat. These issues are biologically important but we do not know yet how to treat them algorithmically.

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.

Thanks is also due to 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 and M. Waterman. A method for fast database search for all k-nucleotide repeats. *Nucleic Acids Research*, 22:4828–4836, 1994.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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*.

[17] 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.

[18] 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	5	2
2	1	100	5	15	2
3	2	30	10	20	2
4	2	30	20	30	2
5	1	10	5	15	1
6	2	100	15	25	2
7	2	10	15	25	1
8	3	100	25	35	2
9	4	100	35	45	2

Table 2: Satellite parameter sets for the exhaustive algorithm.

Chromosome	Parameter set								
	1	2	3	4	5	6	7	8	9
1	23	5	24	46	28	3	400	4	5
2	73	14	61	147	70	42	1228	14	17
3	36	8	33	95	49	12	998	7	8
4	139	26	112	255	202	27	2403	26	32
5	57	12	50	134	62	12	941	11	13
6	29	6	28	55	37	4	493	5	7
7	105	22	93	172	102	15	1725	20	25
8	53	11	44	107	66	8	994	10	12
9	44	13	378	1224	351	77	738	469	10
10	60	11	49	133	66	56	1130	11	14
11	64	13	57	103	78	9	980	12	14
12	101	23	89	252	86	16	2390	19	23
13	86	28	70	222	113	34	2100	16	20
14	73	13	60	118	53	11	1243	14	17
15	101	22	85	252	114	17	2607	19	24
16	86	18	72	187	68	14	1906	17	20

Table 3: Execution times (in seconds) on a Dec Alpha 4000 5/466

Chromosome	Pos. Start	Pos. End	Model	Score 1	Score 2
2	1 486	2 403	GTTGGTAGTTGCAGTAGT	-	215
	68 181	68 551	ATT	134	-
	72 248	72 603	AAC	112	-
	464 053	464 249	GCTTGTGCTTGTGCTTGT	264	164
	541 275	542 091	ATC or ATT	142	122
	780 106	780 769	AAC	108	130
4	149 040	149 241	AAAT	120	77
	161 386	161 848	ATT	62	126
	384 256	385 692	GAAAGTAGTAGAGGAT	454	239
	390 319	390 881	ATT	128	126
	441 152	441 601	TTG	-	126
	778 640	779 123	ATT	182	104
	1 290 850	1 291 293	AAT	116	134
5	1434	2829	AGT	-	156
	142 656	143 393	AAT	148	107
6	4 673	4 957	ACACCC	168	111
7	431 305	431 878	ATT	126	93
	529 844	530 351	ATC	100	131
8	252 652	253 266	ATC	102	116
	365 146	365 725	ATC	96	101
	391 282	392 244	AAT	104	133
	556 785	556 983	GGGTGT	202	120
9	1652	3420	AGT	-	174
	105 188	106 079	AAT	118	152
	334 072	334 882	CTT	-	109
	391 338	392 719	GCAGAAGAGCTTTCAGTGGTAGAGCTGGAG	234	-
	391 338	392 720	CAGAAGAGCTTCAGTAGTAGAG	-	606
	391 338	392 720	GGTACTGGAGCAGAAGAGCTT	-	594
	391 338	392 718	GGAGCAGAAGAGCTTTCAGA	-	567
10	1 635	3 404	AGT	-	174
	469 723	470 067	AAT	108	70
	713 889	715 701	AGT	64	293
	714 438	715 059	GAAGTGGTAG	138	278
11	15 834	16 346	AATAT	-	136
	15 697	16 395	AAT	-	133
	63 864	64 942	CTT	344	268
	240 214	240 647	ATC	66	101
	336 847	337 693	GTT	106	161
	576 031	576 506	CGT	106	100
12	1 647	2 963	AGT	-	126
	11 109	12 131	AC	162	125
	823 330	823 862	AG	128	72
13	86 858	87 418	ATT	228	123
	120 428	121 213	CTT	80	106
	130 001	130 535	CTT	-	113
	587 545	588 026	ATT	140	90
14	454 038	454 801	AAG	-	135
	454 236	454 787	AAGA	-	120
	783 847	784 330	GGGTGT	378	231
15	236 184	236 438	AGCACA	-	137
	236 184	236 379	AGCACA	136	-
	427 781	428 424	CGT	152	131
16	187 226	187 841	CTT	-	132
	252 136	252 618	CTT	126	105
	520 646	521 491	AAT	118	149
	536 587	537 032	CTT	172	104
	650 301	650 928	ATT	170	127
	945 909	947 236	ACT	-	126

Table 4: Satellite models scoring above 100 in at least one of the two scoring systems adopted.