FRAGMENT ASSEMBLY PROJECT (OUTLINE)

Heumann K., Maierl A., Mewes H.W.

Max-Planck-Institut für Biochemie, MIPS,
Am Klopferspitz, 82152 Martinsried,
Germany

e-mail: heumann@mips.embnet.org   mewes@mips.embnet.org

Phone: +49 89 8578 2451
FAX: +49 89 8578 2655

Abstract[1]

In this paper we describe a novel approach for efficient DNA-fragment assembly. Large scale sequencing projects currently require a substantial amount of human interaction to assemble small portions of DNA (around 300 nucleotides, generated by shotgun sequencing techniques [2]) into the underlying contiguous sequence (contig). Computational fragment assembly infers contigs by searching for overlapping subsequences. In its idealized form, assuming error-free fragments and ignoring the existence of dyadic complements, the fragment-assembly-problem has been shown to belong to the class of problems that are NP-hard [3]. Therefore, under practical conditions, this problem is at least NP-hard.

Our approach is based on a formal, pattern-oriented query definition to generate operations supported by a high performance search engine, the Hashed Position Tree (HPT). The HPT is a dynamic, persistent [4] index data structure for large data sets. We currently investigate different fragment assembly schemes realized as propagating patterns. In this context the HPT provides a mechanism for a highly efficient heuristic solution to the NP-hard problem of sequence fragment assembly.

[1] In this paper, we do not present data for the evaluation of the proposed algorithm. Since we received information on the contest organized by the Centre for Discrete Mathematics and Theoretical Computer Science (DIMACS) very late, time was to short to run the test data set until the deadline of the abstract. However, the HPT-based approach appears to be an interesting concept to by applied to the fragment assembly problem, especially for large data sets.

Current software applied to DNA-fragment assembly requires a substantial amount of human interaction [9]. The increase in the output from automatic gel reading devices requires improved algorithms to avoid fragment assembly becoming the rate-limiting step. Not only the large number of reads per day (e.g. 1000 ESTs) but also the size of the DNA contigs being sequenced by 'shotgun' techniques require increased performance in time and space behavior. Recent algorithms address solutions to well defined sub problems are given such as the *All Pair Suffix-Prefix Problem* [5, 6]. Algorithms applied must be scalable and must provide a high degree of automation. The assembled sequence must be correct, i.e. it must correspond to its natural counterpart. The probability of errors as a result of the assembly process should be as low as possible.

Parameters that determine the practical hardness of the problem are the average error rate per fragment, the coverage of target sequence by the fragment set, and the minimal number of contigs that can be generated. In this context we regard the task to remove redundancy from a large data set of independently generated sequences (e.g. ESTs) as a worst case of fragment assembly, where the coverage and the minimal number of contigs are unknown.

Therefore, we approach the fragment assembly problem from two different perspectives:

- As a constraint driven assembly problem: sequence fragments are grouped by their internal overlaps according to their expected coverage. Neighboring fragment sets are matched to generate the superstring.
- As a database redundancy problem: sequence fragments represent a redundant data set. The superstring is the non-redundant representation of the data base.

These two extremes of the problem define the boundaries for the constraints to be applied. Our system should be designed in such a way that it can satisfy these varying constraints in order to generate the optimal solution, i.e. the shortest common superstring or the smallest number of non-redundant sequences.

Both approaches require to limit the exponential extension of the fragment search space generated by brute-force combinatorics and to speed up the sequence comparison itself. Our approach limits the fragment search space by the use of a specialized data structure, the HPT. The HPT evolved from the *position tree* data structure [7,8]. It allows to access strings in the database with a time complexity that is proportional to the length of the query string. This length can be considered as constant. The construction of the position tree from a database of the length n has the time complexity $O(n \log n)$. The HPT is built over the fragment set and its reverse complement. The basics of the HPT were described elsewhere [1].

Organizing the data set into the HPT allows to use a flexible pattern specification to match the different requirements of the assembly process.

A pattern m is defined as a list of pattern components w_i.

$$m = (w_1, w_2, .., w_p)$$

The pattern components w_i represent regions of a sequence fragment with distinct constraints associated. M is the union of all possible patterns m. Sequences s are constructed over a fixed alphabet[2] Σ. W is defined as a set of pattern components w_i.

[2] Σ is the alphabet over the 4 bases {A,T,G,C}.

For example the sequences s_0 = ***ATTTGCTTTTAT*** and s_1 = ***TTTATGCCCC*** constitute a fragment set. They can be described as patterns in the following way:

$m_0 = (w_1, w_2)$ with w_1 = ***ATTTGCT*** and w_2 = ***TTTAT***

$m_1 = (w'_1, w'_2)$ with w'_1 = ***TTTAT*** and w'_2 = ***GCCCC***

In this case the constraint $w_2 = w'_1$ expresses the overlap relationship between s_0 and s_1.

[Example I]

```
m0 =        w1       w2
        ATTTGCT TTTAT
                TTTAT GCCCC
m1 =            w'1     w'2
```

In order to identify patterns m in the data set S, relation f maps a pattern m to sequences s *subset* S.

$f : M \text{ --> } S$

$f : m \text{ --> } s$

Relation f is defined for each pattern component w_i of m as follows:

$f : W \text{ --> } \Sigma^*$

$f : w_i \text{ --> } x_{i,j}$ with[3] $f(w_i) = x_{i,j}$, if $x_{i,j}$ is substring of s *Element* S

else[4] $f(w_i) = \varepsilon$

In order to retrieve specific, constrained pattern components, the relation g selects a set of pattern components w with length l that fulfill a frequency constraint[5].

$g : N\text{x}N \text{ --> } M$

$g : l, h \text{ --> } w$ with $g(l,h) = w$ <=> $|f(w)| >= n$

else $g(l,h) = \varepsilon$.

3 For expressing exact matches one may map $f(w_i)$ to x_i. We will follow this notation to keep the formalism simple.

4 ε denotes the empty word.

5 Relation g is defined for exact matches only in this case.

An efficient implementation of f and g is necessary to match the performance requirements. The following schema shows the decomposition of pattern m:

$$f(m) = s$$

$$f(g(l_1,h),g(l_2,h),...,g(l_p,h)) = s$$

$$f(g(l_1,h))f(g(l_2,h))...f(g(l_p,h)) = s$$

$$f(w_1)f(w_2)...f(w_p) = s$$

$$x_1x_2...x_p = s$$

Taking the example data set s_0 and s_1, only the overlap between two sequences is searched. Therefore the decomposition procedure must yield $s = x_1x_2$. Since the data set consists of two sequences, only two solutions are possible to resolve $f(m) = s$ for all sequences of the data set.

1. $g(l_2,2) = TTTAT$ for maximal l_2 and $x_1 = ATTTGCT$.
2. $g(l_1,2) = TTTAT$ for maximal l_1 and $x'_2 = GCCCC$.

These two solutions define the alternative representations of the overlap relation [see example I]. Once a valid overlap configuration is detected, it can easily be extended by propagating the constraints linearly [see example II]. To realize this propagation pattern m_2 is redefined as $m_2 = f(w'_1,g(l'_2))$ with w'_1 is determined by x_2. Resolving this query will yield a fitting fragment for example $s_3 = GCCCCTCTT$ which can later be used to specify pattern m_3 and so forth.

[Example II]

$$
\begin{array}{lllll}
m_2 = & & x''_1 & & x''_2 \\
& \textit{ATTTGCT} & \textit{TTTAT} & \text{GCCCC} & \textit{TCTT} \\
& & \textit{TTTAT} & \text{GCCCC} & \\
m_1 = & & x'_1 & x'_2 &
\end{array}
$$

Efficient evaluation of f and g is essential to take advantage of the approach. We use the HPT as the underlying search engine for implementing these functions. The specifics of the HPT are described elsewhere [1] .

By the outline given above, fragment assembly algorithms can be expressed relying on pairwise overlap relations. We have chosen this approach to remove redundancy from the EST libraries. However, to achieve a higher quality target sequence, a generalization of this approach allowing to incorporate additional contraints is more favorable. The constraint we apply here is the expected coverage k. A region well covered by fragments will display a pattern of at least k overlaps. This expectation justifies to generalize the approach to select a well defined starting point for the assembly process. The query mechanism allows to express pattern definitions as follows:

$$m_0=(w_1,...,x_{k-1},g(l),\mathrm{w}_{k+1},...,w_{2k-1}) \text{ with } g(l)=\mathrm{w}_k$$

provided the frequency of pattern component w_k is equal to coverage[6] k.

[6] The coverage k determines how many pattern components m has at most.

The x_i (i not equal to k) represent the overlapping edges of the k-fragment set. Similarity is defined with accordance to the expected experimental error rate in the raw data. We allow an equivalent mismatch rate in the w_i pattern components. It is unlikely that a cluster covered by k overlaps incorporates fragments generated from the complementary DNA strand. The subsequence that is covered best by a k-set is selected as the starting point of the assembly process. This is the case for l to become maximal for $g(l) = w_k$, i.e. the longest substring of coverage k.

Provided a starting set has been determined, the HPT is used as an index structure to retrieve any neighboring fragment set. This matching fragment set is retrieved by pattern[7]

$$m_1=(w'_1,\ldots,w'_{k-1},g'(l),w'_{k+1},\ldots,w'_{2k-1})$$
$$\text{with } w'_1 \leftarrow\!\!-\!\!-^{8}\; x_{k+1},\ldots, w'_{k-1} \leftarrow\!\!-\!\!-\; x_{2k-1} \text{ and } g'(l)=w'_k$$

expressing the overlap relationship between pattern m_0 and pattern m_1. This inductive process is repeated for all fragment sets expressed as subsequent pattern specifications where m_j is a construction of m_{j-1}. All fragments that have been incorporated into a contig will be deleted from the HPT during the process of reassembly. As a result, the remaining fragment sets are less likely to contain ambiguous configurations. Since the time complexity of functions f and g depends only on the length of w_i, the reconstruction algorithm operates under optimal conditions linear in time $O(n)$ with n representing the length of the DNA-contig. The overall time requirements is determined by the construction of the HPT. In our current implementation, the HPT is constructed in $O(n \log(n))$ time. Thus, assembly of double stranded DNA-contigs from fragments of coverage k has $O(kn \log(kn))$ overall time complexity.

Within this approach it cannot be guaranteed that the reconstruction algorithm will always generate a solution linear in time. Ambiguities may result for the following reasons:

- No fragment set is available to complete the reassembly. In this case, the reconstruction can be continued by resuming the assembly from a new starting set. With the remaining fragments other non-overlapping contigs can be generated.
- More than one fragment set fits with equal probability to its neighbor. In this case both alternatives are investigated using backtracking to determine the best selection.

In a simulation using randomly generated fragments, the HPT was able to reconstruct a contig of size $n=50{,}000$ (equivalent to shotgun sequencing a complete cosmid). The average size of the fragments was set to 300, coverage of the complete sequence contig was set to $k=5$. This is equivalent to about 1700 fragments. The sequencing error rate was estimated at 2%. Under these conditions the time required for the construction of the HPT overweighed the time required for the reconstruction. The algorithm was implemented on a DEC station 3000. In our simulation, the processing time was less then 1 hour. Currently the system is under investigation using experimental data provided by a sequencing laboratory (generously provided by T. Pohl, GATC, Konstanz). The variation of the coverage k throughout the contig appears to be the most difficult problem for contig assembly from experimental data. In reality, the parameter k varies within the boundaries of an interval. Therefore the fragment assembly may only provide a partial solution. In this case, additional sequencing experiments are mandatory in order to yield the complete contig.

[7] The coverage k determines how many pattern components can be expected at most.

[8] "<--" stands for *determines*. In case of exact matches "<--" is equivalent to "=".

References:

1. Mewes, H.W. and Heumann, K. *Genome Analysis: Pattern Search in Biological Macromolecules.* Combinatorial Pattern Matching; Proceedings; 6th Annual Symposium CPM'95 Espoo, Finland July 1995 Lecture Notes in Computer Science 937, pages 261-285.
2. Maxam, A.M. and W. Gilbert W. *A new method for sequencing DNA.* Proc. Natl. Acad. Sci. USA 74:560-564, 1977.
3. Meier, D., *The complexity of some problems on subsequences and supersequences.* Jour. Assoc. Comput. Mach. 25(2):322-336, 1978.
4. Object Design, Inc., Reference Manual. ObjectStore Release 3.0 Beta. For VAX/VMS Systems. Burlington., 1993.
5. Gusfield, D., Landau, G.M. and Schieber, B. *An efficient algorithm for the All Pairs Suffix-Prefix Problem,* Information Processing Letters 41:181-185, 1992.
6. Gusfield, D. *Faster implementation of a shortest superstring approximation.* Information Processing Letters 51:271-274, 1994.
7. Weiner, P. Linear pattern matching algorithms. In *Conference Record, IEEE 14th Annual Symposium om Switching and Automata Theory,* pages 1-11, 1973.
8. Ukkonen, E. On--line Construction of Suffix Trees. Report A-1993-1, University of Helsinki, Finland, Department of Computer Science, February 1993.
9. Dear, S. Staden, R. *A sequence assembly and editing program for efficient management of large projects.* NAR 19(14):3907-11, 1991.