TRAMP: a software package for generating transposon maps

Eugene Veklerov, Christopher H. Martin and Edward H. Theil

Abstract

TRAMP is a software package for generating transposon maps that are used for DNA sequencing. The package provides a variety of automated tools that can always be overridden by the user. The central part of the package is its selection algorithm that finds the most robust map with the smallest number of inserts. TRAMP has been in daily use by the sequencing team at LBL since it was introduced in the Spring of 1994. It is applicable to any sequencing project utilizing the directed strategy.

Introduction

The goal of the Human Genome Project is to sequence the entire genomes of man and some selected model organisms. This goal will be achievable only if the existing techniques are made more efficient. In particular, the labor costs involved in determining the sequence should be decreased drastically. Ideally, the entire process should be totally automated. Yet, the users need to be able to override automated functions and exercise manual control over every step of the process. This is somewhat analogous to having a program that can run either in a batch mode, or in a debugging mode. The latter mode allows the user to go through every step and make decisions by hand.

The existing techniques do not allow us to achieve this level of automation yet. However, certain parts of the sequencing process have been automated in the sense explained above. This paper deals with one of them, namely with generating transposon maps. There are several sequencing strategies used by different groups. This paper addresses the directed strategy that has been pioneered at the Lawrence Berkeley Laboratory (LBL) and adopted by several other groups. First, we give an overview of the directed strategy (for a detailed description, see Palazzolo et al., 1991; Yoshida et al., 1993). Then, we present the specific part addressed in this paper.

The idea of the directed strategy for DNA sequencing is as follows. The sequencing process is organized hierarchically: P1 clones (the largest unit of the hierarchy) are broken into smaller subclones. The sequencing sites for a subclone are chosen in such a way that they are almost certainly 300-350 base pairs apart. Since the existing sequencing machines, such as the Applied Biosystems, Inc. 373A Sequencer, currently generate around 360-400 reliable base pairs, the priming sites cover the clone with a minimum amount of redundancy. This leads to substantial savings compared to the shotgun strategy. Another important advantage of the directed strategy follows from the prior knowledge of the relative positions of the priming sites. Hence, the assembly process can take advantage of that knowledge, which reduces the number of 'false positive' alignments compared to alignments not aided by this type of information.

In order to be able to choose the priming sites economically, the directed strategy includes the following steps.

1. Physical mapping. For reasons described in Yoshida et al., 1993 and Palazzolo et al., 1991, the directed strategy uses P1 clones as the basis of the physical maps. The average P1 is 80 kb in size. The goal of this step is to identify P1 clones covering a genomic area of interest.

2. DOG tagging. DOGs (Distance, Orientation, Gene-sized) are minimally overlapping subclones that are about 3 kb in size covering a P1 clone. They are iteratively selected using a novel procedure developed at LBL.

3. Transposon mapping. At this step, we select a minimally overlapping set of about 10–12 transposon events, such that any two adjacent events are less than 350 base pairs apart.

4. Sequencing and assembly. Having selected a set of transposon events, we generate two sequences associated with every event and extending in both directions. They provide about 93% of the required sequence of both strands. The generated sequences are currently assembled using the Staden software package (Staden, 1986) and some customized software (Veklerov et al., 1993). If successful, this step reconstructs a DOG subclone. The adjacent DOG subclones are put together using customized software. If some gaps remain, custom oligonucleotides are constructed to determine a sequence filling the gaps. On average, two custom oligonucleotides are required per 3 kb subclone (this provides the typically remaining 7% of the sequence).
The software package described in this paper deals only with Step 3. An efficient biological strategy for dealing with this step has been discussed in the literature (Strathmann et al., 1993). The name of our software, TRAMP, stands for TRAnsposon MaP generator. At this step, we have a number of transposons that are inserted into the target DNA sequence (DOG). The currently used hardware inserts 96 transposon events into the target, of which about 80% are actually located within the DOG. These 50–90 mapped inserts (we will use the terms events and inserts interchangeably) are scattered along the target. The goal of this step is to select a subset of the events that are approximately evenly spaced and, used as the priming sites, are able to reliably cover the DNA fragment.

System and methods

PCR reactions are used to determine the point of insertion of a given event. Specifically, the PCR products determine the distances from the insertion to each end of the target fragment (DOG). Let us call these distances \(L_i\) and \(R_i\) (distances from the \(i\)th insert to the left and right ends). Theoretically, \(L_i + R_i\) should be the same for every \(i\) and equal to the size of the fragment. In reality, these sums vary somewhat. Furthermore, for some valid inserts, either \(L_i\) or \(R_i\) is missing and the biologist may or may not consider these 'one-sided' events in addition to the regular 'two-sided' events.

Therefore, TRAMP is to perform the following tasks:

- determine the best estimate of the length of the DOG;
- determine the best estimates of the positions of the inserts;
- select the best (in some sense) subset of the inserts.

TRAMP is a highly interactive software package written in Smalltalk, which is an object-oriented language (Goldberg, 1989). Smalltalk has convenient facilities for handling graphical interface applications. It comes with a graphical, interactive programming environment. It has good data modelling capabilities and it facilitates producing fast prototypes. The user interface takes advantage of the Model-View-Controller (MVC) architecture that is strongly encouraged by Smalltalk. In our case, the model is a collection of all the inserts together with their attributes. Some of the attributes, such as \(L_i\) and \(R_i\), are raw data, others are computed by TRAMP. This model is presented to the user via three views, each of which has a controller that handles the user's input, such as menu selections and keyboard activity.

First, the user works with the view that plots the sums \(L_i + R_i\) as a histogram (see Figure 1). This view handles the first task, which is determining the best estimate of the length of the DOG. The typical histogram has two peaks related to those inserts that are inside the target clone (we use the words clone and subclone interchangeably) and outside it. The controller provided for this view allows the user to zoom a specific part of the histogram, which enables the user to separate the inserts inside the clone. Then, the user can bring up a menu that provides several computational methods of estimating the total length, such as the average or the median of the histogram. In accordance with our general philosophy, the user can always override automatic functions and enter the length manually and repeat the procedure several times if desired.

Once the length of a clone has been determined, the individual inserts can be placed on the clone. Since the shorter of the two distances \((L_i, R_i)\) is more accurate (due to the nature of the agarose gel electrophoresis system that is used to determine those sizes), it is used to determine the location of the inserts. The other two views in Figure 1 show the locations of the inserts in a graphical and a tabular form, respectively. The graphical view shows the inserts as triangles, the tabular view shows them as the lines of the table. The filled triangles are those which have been selected. The two horizontal bars near the bottom of the graphical view show whether the selected set will result in gaps or not. If there are gaps, the bars show their locations and sizes. Of course, the above is only an estimate, because it is based on estimated locations of the inserts and on the projected lengths of the sequences.

The advantages of the MVC architecture are especially obvious here. Thus, if the user moves the cursor to a given insert and clicks a mouse button, the insert is highlighted in both views. At this point, the user can bring up a menu containing a number of operations with the highlighted insert using either view. For example, the insert may be selected, deselected, or deleted altogether. The menu also allows the user to manually alter the insert's attributes, such as its location. The operation immediately propagates to the other view, because it is actually done in the model and the views are different representations of the same model.

The sequence of actions performed by the user is navigated with the help of the two browsers shown in Figure 2. The top level browser, 'Project Browser', allows the user to choose the subclone for which the transposon map is generated. This browser reflects the hierarchical nature of the directed strategy. When the subclone is chosen, the user presses the button open which opens a second browser called 'Dog Browser'. The latter shows actions appropriate for this level of hierarchy. The left part of the Dog Browser shows actions needed to generate a map, while the right part pertains to the assembly process (not described here). There are several actions shown as oval buttons which can be modified by square radio buttons. For example, the button read data comes
Figure 1. Three views of a transposon map (from top to bottom): a graphical view, a tabular view and a histogram of the clone length.

together with two radio buttons: raw data for initial processing the raw data and saved data for working with previously processed data.

The radio buttons 2 sided only and 1 sided ok enable the user to use only two-sided inserts and both one-sided and two-sided ones, respectively. The change algorithms button gives the user some control over the algorithms, which will be discussed in the next section. The postmortem study button is expected to be used in the future when the user can compare a transposon map with the
actual locations of the inserts that become known after assembling the subclone. Its main use will be to compare different algorithms and tune them to provide the highest accuracy.

Selection algorithms

The selection algorithms, in a broad sense, include these three areas:

- estimating the length of the DOG;
- estimating the positions of the inserts;
- selecting the best (in some sense) subset of the inserts.

We will discuss only the third area because it is the central part of TRAMP. Let us introduce the following notation and definitions.

A selection problem is defined by this collection of items.

\[ L: \text{the length of the clone;} \]
\[ x_1, x_2, \ldots, x_n: \text{the positions of the inserts;} \]
\[ z: \text{the desired distance between the inserts.} \]

The default value of the parameter \( z \) is currently 350 bases but it can be adjusted with the change algorithms button of the Dog Browser. A solution to a selection problem is a subset of inserts, \( x_{i_1}, x_{i_2}, \ldots, x_{i_k} \), such that (assuming that the subset is sorted in increasing order):

\[ x_{i_k} \leq z; \]
\[ x_{i_{k+1}} - x_{i_k} \leq z \text{ for every } k \text{ and } L - x_{i_k} \leq z. \]

In other words, a solution is a subset of the inserts, such that if there are two sequences emanating from each member of the subset in both directions of the length \( z \) each, the sequences will provide a double coverage of the entire clone. In still other words, a solution to a selection problem is simply a transposon map and \( n \) is the map's size. An optimal solution is a solution with the minimum value of \( n \). Typically, the values of \( n \) are of the order of 10–12.

Figure 3. The gaps between the LGA solution and the RGA solution as \( z \) goes from 350 to 325.
A selection problem may or may not have solutions. The latter case (when there are very wide gaps in the positions of the inserts) will not be considered here. In the former case, there may be one or several optimal solutions. Now, let us introduce a 'greedy' algorithm (see, for example, Aho et al., 1983). It simply selects $x_i$ as the right-most insert less than $z$. Then, it selects $x_j$ as the right-most insert less than $x_i + z$, etc. It is easy to show that this greedy algorithm results in an optimal solution. The proof is based on the fact that if $x_i, x_j, ..., x_k$ is a greedy solution and $y_{i_1}, y_{i_2}, ..., y_{i_m}$ is an optimal solution also sorted in increasing order, then $y_k \leq x_k$ for every $k$ (the inequality can be proved by induction on $k$. Hence $n \leq m$. For this reason, we will call the described algorithm the Right Greedy Algorithm (RGA).

The solution derived by RGA has the minimum number of inserts. Yet, it is not necessarily the most desirable solution among the optimal ones. Namely, different optimal solutions have different degrees of robustness. First, let us introduce the notion of robustness on an intuitive level. Assume, for the sake of simplicity, that there are very many original inserts, so that we can select one anywhere. Suppose $L = 1100$; then RGA would select the locations (350, 700, 1050). Although there is no solution resulting in less than three inserts, intuitively, the RGA solution is not the best one, because a large part of the sequence extending from 1050 will be wasted. If, on the other hand, we select three evenly spaced locations (275, 550, 825), the assembly process will be more robust in the following sense. Even though most of the sequences are longer than 350, some are shorter. If a sequence happens to be between 275 and 350, the RGA solution can produce a gap, whereas the evenly spaced solution will have no gaps.

This intuitive notion of robustness can be quantified as follows. Suppose that a selection problem has two solutions of the same size, $S_1$ and $S_2$. Let $a_1$ be the largest interval between two consecutive inserts in $S_1$ and $a_2$ be the largest interval between two consecutive inserts in $S_2$. Then, solution $S_1$ is said to be more robust than $S_2$ if $a_1 \leq a_2$. In other words, the more robust solution guarantees that all the intervals between its consecutive inserts do not exceed a number smaller than a number guaranteed by the less robust solution. (Note that we have chosen the simplest definition of robustness. There may be other definitions quantifying the intuitive notion that a robust solution is a subset of inserts spaced as uniformly as possible within the given initial set of inserts).

Therefore, our problem is to find the most robust solution among all optimal solutions. To do that, let us introduce the Left Greedy Algorithm (LGA) similarly to RGA. Unlike RGA, LGA selects inserts from right to left. Namely, LGA selects $x_i$ as the left-most insert greater than $L - z$. Then, it selects $x_j$ as the left-most insert greater than $x_i - z$, etc. LGA also leads to an optimal solution. The proof is analogous to the proof in the case of RGA.

Thus, all the optimal solutions are between the RGA and LGA solutions. More specifically, if $x_i, x_{i_1}, ..., x_i$ is the RGA solution and $y_{i_1}, y_{i_2}, ..., y_{i_m}$ is the LGA solution (both sorted in increasing order), then any optimal solution $z_{i_1}, z_{i_2}, ..., z_{i_m}$ satisfies these constraints

$$y_k \leq z_k \leq x_k (k = 1, ..., n)$$

In principle, we could approach the problem of finding the most robust map as a problem of integer programming. However, we have implemented a more practical approach. Suppose RGA leads to a solution consisting of $n$ inserts. Then, the LGA solution also has $n$ inserts and all optimal solutions are between the two in the sense of (1). Let us take the next step: reduce the value of $z$ by 1, say from 350 to 349, then to 348, etc. and apply both RGA and LGA at every step. The value of $z$ may remain constant during a few steps. This means that we are able to select inserts in such a way that the maximum distance between successive inserts is guaranteed to be shorter and shorter. But after a certain number of steps, the value of $z$ will finally go up, say, from 10 to 11. Physically, this means that we need more inserts if we want to cover the clone and guarantee a shorter maximum distance between successive inserts.

Let us consider the steps resulting in the same value of $n$. As $z$ goes down, each term of the sequence $x_k$ in equation (1) stays the same or moves to the left, which can be proved by induction. Similarly, each term of $y_k$ stays the same or moves to the right. This means that the set of optimal solutions at every step is a subset of that at the previous step. As these sets get more and more narrow, it is the most robust solutions that survive the process of elimination. This follows from the fact that, as the value of $z$ goes down, we eliminate the solutions with the longest intervals, which is exactly the goal of robustness.

Therefore, we end up with a solution that, first, has the smallest possible size and, second, it is the most robust solution among all the solutions of that size. Strictly speaking, our algorithm may not result in a unique solution, because the RGA and LGA solutions may still be different even for the final value of $z$. However, according to our definition, all these solutions are equally robust.

Let us give a numerical example to illustrate our approach. In this example, we have used a real clone whose estimated value of $L$ was 3441. There were 72 inserts between 1 and $L$. These inserts are shown as bold tick marks along the horizontal lines in Figure 3. The RGA and LGA solutions corresponding to $z = 350$ are

---

**TRAMP: a software package for generating transposon maps**

by guest on 03 May 2018

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/11/2/173/286808
shown in the upper part of Figure 3. Namely, the RGA solutions are the right sides of the 11 rectangles and the LGA solutions are the left sides of the same rectangles. Some rectangles are reduced to vertical lines if \( y_i \) in (1) equals \( x_i \). All the optimal solutions are constrained by these rectangles. The total number of such solutions is the product of 1, 1, 1, 3, 5, 1, 2, 7, 5, 9 and 2 which is 18,900.

As \( z \) goes down, \( n \) remains equal to 11 until \( z \) reaches 327, at which point \( n \) becomes 12. Therefore, we can generate a transposon map with the same number of inserts while guaranteeing that all the adjacent inserts are 328 bases apart or less. If we want them to be 327 or less apart, then we have to use more than 11 inserts. The RGA and LGA solutions corresponding to \( z = 328 \) are shown in the lower part of Figure 3. The total number of such solutions is the product of 1, 1, 1, 2, 3, 1, 1, 1, 2, 1 and 1 which is only 12.

Even though we did not come up with a unique solution, the uncertainty here has been reduced dramatically and only the most robust solutions are left as candidates. At this point, the program can make the final selection either arbitrarily, or by optimizing a certain criterion. Alternatively, the user can make the final selection by hand. The last alternative is not bad, because the user actually has more information about the inserts and can use that information in the decision-making. For example, while the PCR image provides the distances from the insertion to each end of the target fragment, some of the distances have a higher confidence level than others. In principle, these levels of confidence can be used in the selection process.

Although this does not have any ramifications here, the reader can notice that there are no uncertainties in the selection of the first three inserts even in the first pass when \( z = 350 \). This is caused by the fact that the inserts are not distributed uniformly between 1 and \( L \) and there are few inserts in the left part of the clone. The fact that the inserts are not distributed uniformly is rather typical and it follows from the biological techniques currently used.

Post-mortem studies

Another function facilitated by TRAMP is 'post-mortem' studies of the transposon maps. Once a clone is fully assembled, the positions of the inserts are known exactly. Then, they can be compared with the original estimates based on PCR reactions. We have done some preliminary studies in this area. One of the results is shown in Figure 4.

The post-mortem studies will be used in further research aimed at optimizing the algorithms involved in various stages of sequence assembly.

Conclusions

We have described interactive software for generating transposon maps. The package is transportable and easy to use. It provides a variety of automated tools that can always be overridden by the user. The algorithms used by the software can be further tuned to optimize the efficiency of the project. Some of the algorithms are parameterized and the user can easily adjust the parameters at run time. The selection algorithm employed in TRAMP results in a map that has the minimum size and the maximum robustness.

TRAMP has been in daily use by the sequencing team at LBL since it was introduced in the Spring of 1994. With it, the time to process a transposon map was reduced from more than an hour to just two to three minutes. The production goals at LBL call for increasing the number of transposon maps generated each year from several hundred to several thousand. Thus, in the near future, TRAMP should save several man years of effort per year, a good example of how computers can make a significant impact in molecular biology laboratories. In addition to LBL, several other groups have adopted transposon-based sequencing. Hence, the work described in this paper can benefit them too.

Acknowledgements

This work was supported by the Director, Office of Energy Research, Office of Health and Environmental Research, Human Genome Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.
References


Received on July 22, 1994, accepted on December 16, 1994