Computing the Minimal Tiling Path from a Physical Map by Integer Linear Programming

Serdar Bozdag[1], Timothy J Close[2], and Stefano Lonardi[1]

[1] Dept. of Computer Science and Eng., University of California, Riverside, CA
[2] Dept. of Botany and Plant Sciences, University of California, Riverside, CA
{sbozdag,stelo}@cs.ucr.edu, timothy.close@ucr.edu

Abstract. We study the problem of selecting the minimum tiling path (MTP) from a set of clones arranged in a physical map. We formulate the constraints of the MTP problem in a graph theoretical framework, and we derive an optimization problem that is solved via integer linear programming. Experimental results show that when we compare our algorithm to the commonly used software FPC, the MTP produced by our method covers a higher portion of the genome, even using a smaller number of MTP clones. These results suggest that if one would employ the MTP produced by our method instead of FPC's in a clone-by-clone sequencing project, one would reduce by about 12% the sequencing cost.

1 Introduction

A physical map is a linear ordering of a set of clones encompassing one or more chromosomes. Physical maps can be generated by first digesting clones with restriction enzymes and then detecting the clone overlaps by matching the lengths of the fragments produced by digestion. A minimum-cardinality set of overlapping clones that spans the region represented by the physical map is called *minimal tiling path* (MTP).

The problem of determining a good set of MTP clones is crucial step of several genome sequencing projects. For instance, in the sequencing protocol called *clone-by-clone*, first a physical map is constructed, then the MTP is computed and finally, the clones in the MTP are sequenced one by one [9]. The clone-by-clone sequencing method has been used to sequence several genomes including *A. thaliana* [15] and *H. sapiens* [12] among others. Also, in several recent whole-genome shotgun sequencing projects, the MTP obtained from a physical map has been employed to validate and improve the quality of sequence assembly [23]. This validation step has been used, for example in the assembly of *M. musculus* [10], *R. norvegicus* [13], and *G. gallus* [18].

With the introduction of next-generation sequencing machines (454, Solexa/Illumina, and ABI SOLiD) we expect the MTP computation to become an essential step in *de novo* sequencing projects of eukaryotic genomes. Next-gen sequencing technology produces massive amount of very short reads (about 250bps for 454, 35bps for Illumina and SOLiD) [4] and therefore the *de novo* assembly of the whole eukaryotic genomes is extremely challenging [17]. Arguably, the only feasible method at this time is a clone-by-clone approach, where each clone in the MTP is sequenced using next-gen technology, and the assembly is resolved separately on each clone (see [17,21,25] and references therein).

K.A. Crandall and J. Lagergren (Eds.): WABI 2008, LNBI 5251, pp. 148–161, 2008.

If the exact locations of all clones in the physical map were known, computing its MTP would be straightforward; simply select the set of clones in the shortest path from the leftmost clone to the rightmost clone in the interval graph representing all the clones. This, however, is not a realistic solution. The noise in the fingerprinting data makes it impossible to build a perfect map. As a consequence, determining the minimal tiling path becomes a challenging computational problem. On one hand, a method that tends to select more clones as MTP might include many *redundant clones* (i.e., clones that do not provide additional coverage) and therefore it would waste time and money later in sequencing. On the other hand, an approach that tries to reduce the number of MTP clones may introduce gaps between the clones in some of the contigs, and thus reduce the coverage.

Although the problem of computing MTP has been studied extensively in the literature (see e.g. [22,15]), in practice there is only commonly used software tool, namely FingerPrinted Contigs (FPC) [7]. FPC provides three methods to compute an MTP, but only one uses solely restriction fingerprint data (hereafter called FPC-MTP). FPC-MTP computes the approximate overlap between clones in the contig and validates each overlap by using three extra clones, a *spanner* that verifies the shared fragments of the pair and two flanking clones that extend to the left and right of the pair and confirm fragments in the pair that are not confirmed by the spanner clone. Once the verification of the overlapping fragments between each clone pair is completed, a shortest path algorithm is used to find the minimal tiling path [2].

FPC-MTP's algorithm is quite good, but can be improved; our experimental results show that FPC-MTP is significantly distant from the optimal[1] MTP. In general, FPC-MTP selects fewer clones than necessary which in turns reduces the overall coverage. By changing parameters one can increase the coverage, but this comes at the cost of introducing many redundant clones. This limitation of FPC-MTP can be attributed to the fact that it checks the clone positions as an overall constraint when computing the MTP [2]. However, the positions of clones in contigs are known to be not very reliable [16].

Our contribution. We propose a new algorithm, called FMTP, that computes the MTP of a physical map based purely on restriction fingerprint data (and the contigs). In other words, our algorithm completely ignores the ordering of clones obtained by the physical map algorithm.

FMTP first computes a *preliminary* MTP by selecting the smallest set of clones that covers the genomic region that is covered by all clones in the contig. The problem of computing the preliminary MTP set is formulated in a combinatorial optimization framework as an Integer Linear Program (ILP). The preliminary MTP set may contain redundant clones. In the second phase, FMTP orders the clones in the preliminary MTP and computes the final MTP by using a shortest path algorithm.

We carried out an extensive set of experiments on the physical map of rice and barley. For the former dataset, the actual coordinates for the clones are known and therefore we could measure the accuracy of our algorithm. The experimental results show that the set of MTP clones computed by FMTP on the physical map for rice has higher coverage than the one produced by FPC (using approximately the same number of clones overall). This suggests that a larger portion of the genome could be obtained at the same cost when

[1] The optimal MTP is the one that we could compute if we knew the coordinates of all the clones.

FMTP is used instead of FPC. Our experimental results also show that if one fixes a target coverage, FMTP produces MTPs with about 12% fewer clones than MTPs produced by FPC. This suggests that our tool could reduce sequencing costs by about 12%.

2 Basic Concepts

We use the term *clone fragment* (or *fragment*) to indicate a portion of a clone obtained by digesting it with a restriction enzyme. Let $b(u)$ be the size for clone fragment u. We say that two clone fragments u and v *match* if their corresponding sizes are within the *tolerance* T, i.e., if $|b(u) - b(v)| \leq T$. The tolerance parameter depends on the fingerprinting method, thus it should match the one used in the construction of the physical map [19]. Given a fragment u, let $N(u)$ be the name of the clone which u belongs to. This notion can be extended to a set of fragments as follows. Given a set of fragments U, $N(U)$ represents the set of all clones that contain at least one fragment in U.

We declare two clones c_i and c_j to be *overlapping* if their Sulston score S is lower than or equal to a user-defined *cutoff* threshold C, i.e., $S(c_i, c_j) \leq C$. The Sulston score measures the probability that two clones share a given number of restriction fragments by chance according to a binomial probability distribution [20]. FPC has an analogous cutoff parameter to build the physical map [19].

A *matching fragment graph* (MFG) is a weighted undirected graph $G = (V, E)$ in which V represents the set of all clone fragments and an edge $(u, v) \in E$ exists if fragment u matches fragment v. The weight on the edge $(u, v) \in E$ is defined as the negative logarithm of $S(N(u), N(v))$. In the ideal MFG, each connected component of the matching fragment graph G should correspond to a unique region in the target genome. Unfortunately, this in not realistic due to noise in the fingerprinting data and falsely matching fragments.

Given a connected component $U \subseteq V$ of G, we call $N(U)$ a *connected component cloneset* (or *cloneset* if it is clear from the context). We say that a clone c *covers* a connected component U of G if it belongs to the corresponding cloneset, i.e., $c \in N(U)$. Given an MFG G, the *minimal tiling path* of G is the smallest set M of clones that cover all connected components of G, i.e., $M \cap N(U) \neq \emptyset$ for all connected components U of G.

A toy example of an MFG is shown in Figure 1-LEFT. Each node is labeled as <clone name>-<fragment size>-<copy number of the fragment>. Clones `K`, `L`, `M`, `N`, and `O` are overlapping. Shared fragments are represented by the connected component of the MFG (namely, fragments of approximate size of 1870, 1805, 1255, and 1400 bases respectively). For example, given the connected component U = {L-1803-1, M-1807-1, N-1802-1}, the cloneset $N(U)$ is {`L`,`M`,`N`}. The minimum tiling path is {`L`,`O`} because by selecting these two clones we cover all the connected components of the graph.

3 Methods

FMTP consists of two modules, namely MTP-ILP and MTP-MST. The module MTP-ILP computes a preliminary MTP by solving an integer linear program. The objective of MTP-ILP is to cover all connected components in the MFG using the smallest possible set of clones. The module MTP-MST computes the final MTP based on an overlap

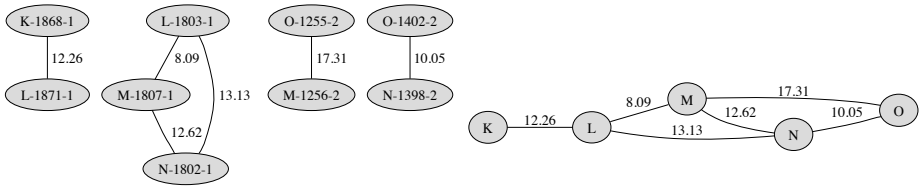

Fig. 1. (LEFT) An example of a matching fragment graph. Each node is a clone fragment and it is labeled as the <clone name>-<fragment size>-<copy number of the fragment>. Each edge is weighted by the negative logarithm of the Sulston score between the clones that contain the incident node fragments. (RIGHT) Overlap graph of clones in this MFG. There is an edge between two clones if their fragments are together in at least one connected component in the MFG.

graph. The goal of MTP-MST is first to order the clones in the preliminary MTP and then run a shortest path algorithm to compute the final MTP. In both modules, an MFG is constructed by performing the steps described below.

3.1 Constructing the MFG

For each contig, FMTP performs a pairwise alignment between all clones in that contig based on their restriction fingerprint data. The alignment produces the initial MFG. Then, some of the false positive edges (i.e. falsely matched fragments) are removed and the final MFG is constructed, as described below in more details.

Preprocessing: burying clones. The purpose of this preprocessing step is to reduce the problem size by discarding clones that are almost completely contained in another clone. A clone is defined *buried* if $B\%$ or more of its fragments are matching fragments of another clone, where B is a user-defined parameter. According to our experiments, B should be at least 80 to avoid false positive buried clones. If two clones can be buried into each other, the smaller of the two (i.e., the one with fewer fragments) is buried into the other one. FPC also buries clones during the process of building the physical map, but it does not discard them during MTP computation.

Building the preliminary MFG. First, we align the fingerprint data for each pair of overlapping clones. For each clone pair (c_i, c_j) for which $S(c_i, c_j) \leq C$, we build a bipartite graph $G_{i,j} = (L_i \cup R_j, E_{i,j})$, where L_i and R_j consist of the fragments of c_i and c_j, respectively, and $E_{i,j} = \{(u, v) | u \in L_i, v \in R_j$ such that $|b(u) - b(v)| \leq T\}$.

In order to align clones c_i and c_j, we search for the maximum bipartite matching in $G_{i,j}$. The matching of maximum cardinality is found by solving max flow on the corresponding flow network [6]. Let $M_{i,j}$ be set of matched edges.

For all clone pairs c_i and c_j for which $S(c_i, c_j) \leq C$, the matching edges in $M_{i,j}$ are used to create the (preliminary) matching fragment graph G. Specifically, for each edge $(u, v) \in M_{i,j}$, nodes u, v and edge (u, v) are added to G (unless they have been already added). The weight of (u, v) is set to be the negative logarithm of the Sulston score between clone c_i and clone c_j.

The objective of the bipartite matching is to attempt to group together clone fragments that are located at the same location on the genome. Because of the noise in the

fingerprint data, some of the matched fragments might not represent the same region in the target genome. In the following steps, we try to eliminate as many false matches as possible.

MFG pruning. In this step, some of the components of G that might represent more than one unique region in the target genome are split. Specifically, we examine all the connected components of G and mark the ones that satisfy at least one of the following conditions as *candidates*.

1. Extra fragment: The connected component contains multiple fragments of a clone.
2. Unmatched fragments: The difference between the length of the second shortest and the second longest fragment in the connected component is more than the tolerance value T. We ignore the shortest and the longest fragments to allow two outliers per component.
3. Weak overlap: The connected component contains at least one pair of clones that are very unlikely to overlap (i.e., have Sulston score of at least 1e-2.5 for MTP-ILP or 1e-1 for MTP-MST).

For each candidate component, a min-cut algorithm is used to partition it by removing the weakest set of edges (i.e., minimum total edge weight) [11]. After a component is partitioned into two subgraphs, we check both subgraphs against the three conditions. If at least one is satisfied, we partition that component again. This iterative process terminates as soon as there are no more components that need to be split.

Elimination of spurious components. In order to introduce the notion of spurious components, let us assume for the time being that the exact locations of the clones in a contig are known. Let U be a connected component of G, and let $N(U)$ be the corresponding connected component cloneset. We call U *spurious* if the overlapping region for the clones in $N(U)$ is spanned by another clone in the contig. An example is illustrated in the Appendix.

This step is crucial to reduce the number of redundant clones in the preliminary MTP produced by MTP-ILP. Recall that MTP-ILP selects the smallest set of clones that cover each connected component in the MFG. If the spurious components are not removed, at least one clone in each spurious component is added to the MTP without contributing to the overall coverage.

Since the exact ordering of the clones is not available, we employ a probabilistic method to detect spurious components. More specifically, we compute the probability that a connected component is observed purely by chance. If this probability is high then we mark the component as spurious. We expect the number of spurious components to be low, since missing fragments or extra fragments are unlikely to occur frequently for a specific cloneset.

In our model, we assume that fragment sizes are distributed uniformly, and that the probability that two clones share a fragment is $P = 2T/gellen$ [19], where T is the tolerance and $gellen$ is the number of possible fragment size values. The probability that two clones share f fragments is P^f and the probability that c clones share f fragments is $P^{(c-1)f}$. This suggests that the probability that a connected component U is observed

in the MFG is $P^{(|U|-1)f}$, where f is number of times that $N(U)$ forms a connected component. In our algorithm, if $(|U|-1)f$ is lower than a threshold Q then we mark U as spurious. Spurious components are removed from the MFG permanently.

Removing clones that are buried into multiple clones. Recall that the goal of the first step in the construction of the MFG is to bury clones into other clones to reduce the problem size. In this step, we reduce the problem size further by removing clones that are buried into multiple clones. For example, in Figure 3 in the Appendix, clone C is buried into B ∪ D, thus C can be removed.

For each clone in a contig, we compute the ratio between the number of its fragments in the MFG and the total number of its fragments. If at least $B'\%$ of its fragments are already in the MFG, then we mark that clone buried. However, there is a complication. It is possible that two clones are *mutually buried* and therefore we cannot remove both. In order to solve this problem, we first identify all possible candidates that can be buried into multiple clones. Then, we examine each connected component U of G, and save $N(U)$ in a list L if all clones in $N(U)$ are candidates to be buried.

All candidates that do not exist in any $N(U) \in L$ can be buried. However, we need to make sure that at least one clone in each $N(U) \in L$ is not buried, otherwise the region in the target genome that is represented by U is not covered. So the task at hand is to remove as many candidate clones as possible with the constraint that at least one clone survives from each cloneset $N(U) \in L$. This problem is called *minimum hitting set* and it is NP-complete (polynomial reduction from the vertex cover problem [8]). Here, we solve it sub-optimally using a greedy approach [8]. At each iteration, we (1) select the clone c that occurs in the maximum number of clone sets in L, (2) remove all clone sets that contain c, (3) save c into minimum hitting set H, and (4) repeat. The iterative process terminates when the list L is empty.

Buried clones selected in this way are removed from the MFG permanently. If this removal introduces components with only one node, they are also removed from the MFG permanently. The algorithm is summarized as Algorithm 1 in the Appendix.

Adding mandatory clones to the MFG. Clones that have to be selected as MTP clones are called *mandatory* clones. More specifically, if at least 50% of the fragments of a clone are not present in the MFG, then that clone is marked as mandatory. When a clone is marked as mandatory, it is immediately stored in the MTP and ignored for further analysis by removing all of its fragments from the MFG.

3.2 Solving the MTP Via ILP

Recall that the MTP can be computed by selecting the smallest set of clones that cover all connected components of the MFG. This problem is a special case of the *minimum hitting set* problem. Although this problem is NP-complete on general graphs [8], it can be solved optimally in polynomial time on overlap (or interval) graphs [3]. Here, we solve this problem optimally by expressing it as an integer linear program (ILP), but first we remove any connected component from the MFG that does not affect the solution.

We reduce the problem size by removing some of the connected components of G according to the following Lemma (the proof is immediate). This simplification step drastically reduces the execution time required to solve the ILP (about 200-fold for rice).

Lemma 1. *Let U, V be two connected components of G, such that $N(V) \subset N(U)$ and G' is a graph obtained from G by removing U. If M is an MTP for G' then M is an MTP for G.*

The MTP problem can be expressed as follows

$$\textit{Minimize} \quad \sum_{c \in N(V)} X[c]$$

$$\textit{Subject to} \quad \sum_{c \in N(V)} X[c].CC[c, U] \geq 1 \quad \forall U \in G$$

$$X[c] \in \{0, 1\} \quad \forall c \in N(V)$$

The coefficient $CC[c, U]$ is 1 if clone $c \in N(V)$ has a fragment f that belongs to the connected component U of G, 0 otherwise. The integer variable $X[c]$ is 1 if $c \in N(V)$ is selected, 0 otherwise. When c is selected, all connected components U of G such that $c \in N(U)$ are covered.

We generate one ILP for each contig. The number of variables in each ILP is $|N(V)|$ (i.e., number of clones that contain fragments in the MFG). We do not need to relax the ILP to a linear program, because the problem size is small enough that can be solved optimally using GNU Linear Programming Package (GLPK). The clones in each solution set are then merged in the preliminary MTP. This preliminary MTP is very likely to contain several redundant clones and need to be processed further.

3.3 Solving the MTP Via MST

From now on, all the clones in the physical map that do not belong to the preliminary MTP are disregarded. A new MFG $G = (V, E)$ is constructed only on the clones in the preliminary MTP. Then, an overlap graph is built from G and the minimum spanning tree (MST) of the overlap graph is computed to order the clones. Finally, the shortest path from the first clone to the last clone in the ordering is computed. Clones on this path constitute the final MTP. Here are the details.

First, MTP-MST attempts to order the clones in the preliminary MTP. For this purpose, a weighted overlap graph $G_O = (V_O, E_O)$ is constructed for each contig, where V_O is the set of preliminary MTP clones in each contig and $E_O = \{(u, v)|$ There exists a connected component U of G such that $\{u, v\} \subseteq N(U)\}$. The weight of edge $(u, v) \in E_O$ is again the negative logarithm of the Sulston score between u and v.

Figure 1-RIGHT shows the overlap graph for the MFG in Figure 1-LEFT. For example, there is an edge between clones K and L because their fragments occur together in at least one connected component in the MFG (see Figure 1-LEFT).

In [24] we established that under some assumptions on the metrics (which are also met here), the MST of an edge-weighted overlap graph gives an ordering of nodes in the graph. One of the assumptions is that edge weights are proportional to the overlap size. Although the correlation between Sulston score and overlap size is not perfect, the MST still gives very accurate ordering because the clones come from the preliminary

MTP and not the original problem. Recall that in the preliminary MTP a clone is not expected to overlap to many clones with similar overlap size.

According to our experiments, the MST of G_O is usually a path. When the MST is not a path, the relative ordering of some of the clones may not be determined. However, this is not a serious problem if we can detect a pair of overlapping clones that cover the group of clones whose order is undetermined. Because of this overlap, clones with unknown relative ordering do not need to be in the MTP, hence they can be discarded in the analysis.

Once the MST of G_O is computed, we pick the longest path $\mathbb{P}$ in the tree. If there is more than one such path, we select the path with the smallest total weight. The rationale is to minimize the total overlap size between consecutive clones, and thus select the path with higher coverage. The clones in $\mathbb{P}$ cover almost the whole contig, but they may not be an MTP. In order to find the MTP, the path $\mathbb{P}$ must be augmented with high confidence overlap edges and a shortest path must be found.

To minimize the possibility of adding false negative and false positive edges to $\mathbb{P}$, we use several criteria based on the Sulston score and the MFG. The details of this algorithm are shown in Algorithm 2 in the Appendix. After augmenting $\mathbb{P}$, the shortest path from u_1 to $u_{|\mathbb{P}|}$ (in terms of number of hops) is computed. All nodes in this path are chosen as the MTP clones of this contig.

4 Experimental Results

We used the genomic data of two plants, namely barley and rice, to compare our software to FPC-MTP (hereafter called FPC to be clearer). We ran FPC and FMTP on the physical maps of rice and barley and obtained their MTPs. The physical maps were obtained using FPC and our compartmentalized method [5] on real restriction fingerprints of BACs. Restriction fingerprint data for rice and barley BACs were obtained from AGCoL [1] and our NSF-sponsored project [14], respectively.

The physical map that we used for rice contains 22,474 clones, 1,937 contigs and 1,290 singletons. The publicly available rice physical map [1] is a superset of our map because we selected only clones that could be uniquely mapped to the rice genome. Barley physical map contains 72,052 clones, 10,794 contigs, and 10,598 singletons.

Since the barley genome has not been fully sequenced yet, it is not possible to equivalently assess the quality of the barley MTP. However, we obtained genomic coordinates for all rice BACs in the physical map by locating their BAC end sequences (BESs) on the genome. Details of this procedure were given in [5]. Before we present the experimental results, we describe how to compute the best and the "average" MTP, which metrics are used to evaluate the quality of the MTP, and the parameters used.

4.1 Optimal and Random MTP

One of the first questions we asked ourselves was "Given a physical map, what is the best (and the average) coverage the MTP can achieve?". If the coordinates of the clones are known, then we can compute the optimal MTP as follows. Given a contig, we compute the optimal MTP by first building a directed interval graph where each node is

a clone and there is an edge between two clones if their coordinates are overlapping. Buried clones are obviously disregarded. Then, we compute the shortest (unweighted) path from the leftmost clone to the rightmost clone. The clones in the path constitute the optimal MTP. We record the number of clones of the MTP and its genome coverage.

In order to have a fair comparison between the computed MTP and the optimal MTP, we also limit the number of MTP clones that can be selected as optimal MTP. The number of optimal MTP clones in a contig is bounded by the number of computed MTP clones in that contig.

If the optimal MTP is the best coverage we can hope for, the random MTP is the one we could achieve if we had no particular strategy. For the random MTP, we select a set of clones for each contig at random, according to a uniform distribution. The total number of MTP clones selected is matched to the total number of MTP clones computed by either FPC or FMTP. The number of MTP clones that are selected for each contig is determined by distributing the total of available MTP clones among the contigs proportional to their size.

4.2 Metrics

Several metrics have been devised to evaluate the quality of the MTP. Due to lack of space, we can only describe one.

Global and contig-wise coverage. The *global coverage* of an MTP is the ratio between the total length of the region spanned by all MTP clones and the genome size.

The *contig-wise* coverage of a contig is the ratio between the coverage of MTP clones and the coverage of all clones in the contig. The contig-wise coverage is computed for each contig and then an overall score is computed as the weighted average of contig-wise coverage, where the weight is the number of MTP clones in each contig. Although contig-wise coverage appears very similar to global coverage, it may produce different results when a region in the genome is covered by multiple contigs.

4.3 Parameters

Both FPC and FMTP have six parameters. Depending on the fingerprinting method (i.e., agarose or High Information Content Fingerprinting (HICF)), FMTP provides default values for its parameters. Using values for these parameters close to the defaults is crucial to obtain good performance. For example, the cutoff parameter C should be changed slightly. By default, MTP-ILP uses a low C value (1e-10 for agarose or 1e-40 for HICF). Since MTP-ILP processes the original contigs which usually contain many clones, a higher value of C would introduce many false positive overlaps. On the other hand, to detect shorter overlaps, MTP-MST uses a high C value (1e-2 for agarose or 1e-10 for HICF).

We have generated a large number of MTPs using both tools with several parameter sets, however we only recorded the best possible MTP for a given size (i.e., number of clones). If we obtained two MTPs M_i and M_j using different parameter values, if the size of M_i is greater than the size of M_j then the coverage of M_i must be greater than coverage of M_j, or otherwise M_i is disregarded. As a result, in the experimental results as the size of the MTP increases, the coverage increases monotonically.

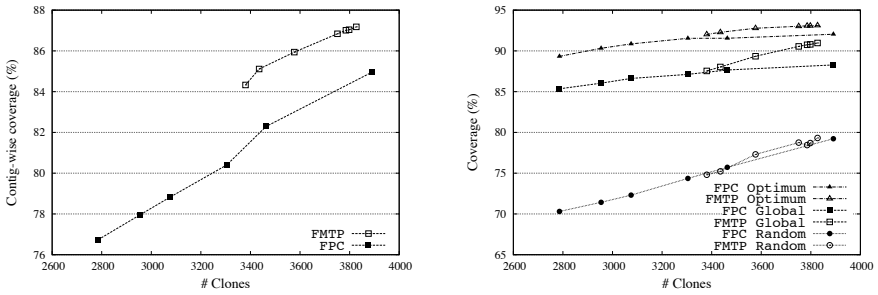

Fig. 2. Comparing the MTPs generated by FMTP and FPC with several parameter sets. Each point in the graph represents a unique MTP. (LEFT) Contig-wise coverage, (RIGHT) Optimal, global, and random coverage.

In order to have a fair comparison between FMTP and FPC, we used the same B and T values used by FPC when building the physical map ($B = 90$, $T = 7$ for rice, and $T = 3$ for barley). We set the C for the MTP-MST to values between 9e-2 and 5e-4. For all other parameters, we used the default values (Q=3, B'=80, C'=1e-2.5 for MTP-ILP, 1e-1 for MTP-MST, C=1e-10 for MTP-ILP).

4.4 Evaluation Results

The graphs summarizing the results are shown in Figure 2. Each point in the graphs represents an MTP. Figure 2-LEFT shows the contig-wise coverage as a function of the number of MTP clones; Figure 2-RIGHT illustrates optimal, global, and random coverage of the MTPs as a function of the MTP size.

First, observe that when all clones in the rice contigs are selected as MTP clones, the global coverage is only 94.45% of the genome. As shown in Figure 2-LEFT and RIGHT, FMTP produces MTPs with significantly better contig-wise and global coverage than FPC, sometimes even with fewer clones. For instance, the highest possible contig-wise coverage that we were able to obtain by using FPC is 84.96%, whereas FMTP's is 85.11% with about 460 (12%) less clones. This would imply 12% reduction in the sequencing costs. Also, global coverage of MTPs produced by FMTP converges to the optimum coverage much faster than FPC.

The number of redundant clones and gaps produced by FPC and FMTP are almost identical and very small. The average overlap size between consecutive clones is smaller in FMTP than FPC (data not shown). This explains the difference in coverages in Figure 2.

Another interesting observation can be made by comparing the optimal coverage in Figure 2-RIGHT. The optimal MTP for FMTP has higher coverage and has a smaller number of clones than the optimal MTP of FPC. Recall that the optimal coverage is computed by selecting a set of clones for each contig that covers the widest possible region. A higher coverage (even when the number of clones is smaller) suggests that FMTP selects relatively more MTP clones from the "big" contigs than FPC.

We ran FMTP and FPC on the barley physical map generated by our group at the University of California, Riverside and several other institutions [14]. FPC generated

MTPs that contain between 11,000 and 21,000 clones. When default values are used, FMTP generated MTPs that contain about 18,000 clones.

In terms of running time, FMTP and FPC are comparable. Both tools compute MTP in a couple of hours.

5 Conclusions

We presented a set of novel algorithms to compute the MTP of a physical map by using a two-step approach. In the first step, we used a stringent threshold to reduce the problem size by generating a preliminary MTP without compromising the coverage of the contigs. In the second step, we attempted to order the clones in the preliminary MTP by computing MST of an overlap graph. Then, we ran a shortest path algorithm to compute the MTP. Our experimental results show that our method generates MTPs with significantly higher coverage than the most commonly used software FPC, even using a smaller number of MTP clones. Our experimental results also show that FMTP could reduce substantially the cost of clone-by-clone sequencing projects.

Acknowledgements

The authors would like to thank Prof. Carol Soderlund, Dr. William Nelson, and the anonymous reviewers for insightful comments that helped improving the manuscript. This project was supported in part by NSF CAREER IIS-0447773 and NSF DBI-0321756.

References

[1] Rice physical map data, `ftp://ftp.genome.arizona.edu/pub/fpc/rice/`
[2] FPC-MTP tutorial webpage, `http://www.agcol.arizona.edu/software/fpc/userGuide/mtpdemo/`
[3] Asano, T.: Dynamic programming on intervals. In: Hsu, W.-L., Lee, R.C.T. (eds.) ISA 1991. LNCS, vol. 557, pp. 199–207. Springer, Heidelberg (1991)
[4] Bentley, D.R.: Whole-genome re-sequencing. Curr. Opin. Genet. Dev. 16(6), 545–552 (2006)
[5] Bozdag, S., Close, T.J., Lonardi, S.: A compartmentalized approach to the assembly of physical maps. In: Proceedings of BIBE, pp. 218–225 (2007)
[6] Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM 19(2), 248–264 (1972)
[7] Engler, F.W., Hatfield, J., et al.: Locating sequence on FPC maps and selecting a minimal tiling path. Genome Res. 13(9), 2152–2163 (2003)
[8] Garey, M.R., Johnson, D.S.: Computers and intractability; a guide to the theory of NP-completeness. W.H. Freeman, New York (1979)
[9] Green, E.: Strategies for the Systematic Sequencing of Complex Genomes. Nature Reviews Genetics 2, 573–583 (2001)
[10] Gregory, S., Sekhon, M., et al.: A physical map of the mouse genome. Nature 418, 743–750 (2002)
[11] Hao, J., Orlin, J.B.: A faster algorithm for finding the minimum cut in a directed graph. J. Algorithms 17(3), 424–446 (1994)

[12] International Human Genome Sequencing Consortium. A physical map of the human genome. Nature 409, 934–941 (2001)
[13] Krzywinski, M., Wallis, J., et al.: Integrated and Sequence-Ordered BAC- and YAC-Based Physical Maps for the Rat Genome. Genome Res. 14(4), 766–779 (2004)
[14] Madishetty, K., Condamine, P., et al.: Towards a physical map of the barley "gene space" (in preparation, 2008)
[15] Marra, M., Kucaba, T., et al.: A map for sequence analysis of the Arabidopsis thaliana genome. Nat. Genet. 22(3), 265–270 (1999)
[16] Nelson, W., Soderlund, C.: Software for restriction fragment physical maps. In: The Handbook of Genome Mapping: Genetic and Physical Mapping, pp. 285–306 (2005)
[17] Pop, M., Salzberg, S.: Bioinformatics challenges of new sequencing technology. Trends Genet. 24(3), 142–149 (2008)
[18] Ren, C., Lee, M., et al.: A BAC-Based Physical Map of the Chicken Genome. Genome Res. 13(12), 2754–2758 (2003)
[19] Soderlund, C., Humphray, S., et al.: Contigs Built with Fingerprints, Markers, and FPC V4.7. Genome Res. 10(11), 1772–1787 (2000)
[20] Sulston, J., Mallett, F., et al.: Software for genome mapping by fingerprinting techniques. Comput. Appl. Biosci. 4(1), 125–132 (1988)
[21] Sundquist, A., Ronaghi, M., et al.: Whole-genome sequencing and assembly with high-throughput, short-read technologies. PLoS ONE 2(5), 484 (2007)
[22] Venter, J.C., Smith, H.O., Hood, L.: A new strategy for genome sequencing. Nature 381(6581), 364–366 (1996)
[23] Warren, R.L., Varabei, D., et al.: Physical map-assisted whole-genome shotgun sequence assemblies. Genome Res. 16(6), 768–775 (2006)
[24] Wu, Y., Bhat, P., et al.: Efficient and accurate construction of genetic linkage maps from noisy and missing genotyping data. In: Proceedings of WABI, pp. 395–406 (2007)
[25] Zerbino, D., Birney, E.: Velvet: Algorithms for de novo short read assembly using de bruijn graphs. Genome Res. 18(5), 821–829 (2008)

Appendix

An Example of a Spurious Component

In Figure 3, an illustration of five clones in a contig are shown based on their real coordinates in the genome. Suppose now that the MFG of this contig contains a connected component that contains fragments only from clones `B` and `D`. That component is spurious (i.e., does not represent a region in the target genome) because the overlap between clones `B` and `D` is completely covered by clone `C`. Spurious components arise for two reasons, namely missing fragments (in clone `C` in the example) or extra fragments (in clone `B` or `D` in the example).

Fig. 3. A layout of five clones in a contig based on their actual genome coordinates

Algorithm 1

Input: Set of clones $\mathbb{C}$ in the contig
Input: $G = (V, E)$ {MFG of $\mathbb{C}$}
Output: G {Graph obtained by removing clones that are buried into multiple clones}
A={} {List of candidate clones}
for all clones $c \in \mathbb{C}$ **do**
  $m_c = 0$ {number of times c has a fragment in G}
  for all connected components $U \subseteq V$ **do**
    if $c \in N(U)$ **then**
      $m_c = m_c + 1$
  $r_c = m_c/|c|$ {$|c|$ is the number of fragments in c}
  if $r_c \geq B'$ **then**
    Add c to A
$L = \{\}$ {Set of clone sets}
for all connected components $U \subseteq V$ **do**
  if $N(U) \subseteq A$ **then**
    $L.insert(N(U))$
H =Solve Minimum Hitting Set(L)
for all clones $c \in A - H$ **do**
  for all nodes $v \in V$ **do**
    if $N(v) = c$ **then**
      Remove v from G
Remove connected components of size 1 from G

Algorithm 2

Input: The longest path $\mathbb{P}$ from the MST of G_O
Input: Clones $u_i \in \mathbb{P}$, $1 \leq i \leq |\mathbb{P}|$, where i is the order of u
Input: $G = (V, E)$ {MFG of the preliminary MTP clones in the contig}
Output: Augmented path $\mathbb{P}$
for $d = 2$ to $|\mathbb{P}| - 1$ **do**
  for $i = 1$ to $|\mathbb{P}| - d$ **do**
    Check if $S(u_i, u_{i+d-1}) \leq C$
    Check if $S(u_{i+1}, u_{i+d}) \leq C$
    Check if there exists a connected component U of G such that $\bigcup_{j=i}^{i+d} u_j \subseteq U$
    Check if $S(u_i, u_{i+d}) \leq C$
    if All conditions are true **then**
      Add (u_i, u_{i+d}) to $\mathbb{P}$.

Pseudocode for Removal of Clones that are Buried into Multiple Clones

A sketch of the algorithm that removes clones that are buried into multiple clones is presented as Algorithm 1.

Pseudocode for Clone Overlap Detection

A sketch of the algorithm that detects clone overlap is presented as Algorithm 2.

At each iteration four conditions are checked to determine if u_i and u_{i+d} are overlapping where u_i, $1 \leq i \leq |\mathbb{P}|$, is the *ith* clone in $\mathbb{P}$. All conditions have to be true to add the edge (u_i, u_{i+d}) to $\mathbb{P}$.

In line 7 and 8 of Algorithm 2 we check whether the clone pairs u_i, u_{i+d-1} and u_{i+1}, u_{i+d} are overlapping. Obviously, if at least one of these pairs do not overlap then u_i and u_{i+d} cannot be overlapping (assuming that no clone is completely contained in another clone). In line 9, we check if clones $u_i, u_{i+1}, \ldots, u_{i+d}$ have fragments together in at least one connected component of G. If u_i and u_{i+d} are overlapping then $u_i, u_{i+1}, \ldots, u_{i+d}$ have to be overlapping, and therefore they should share at least one fragment in G. At the end, we check if $S(u_i, u_{i+d}) \leq C$.