The results of applying RSDDM to simulated MRE maps are presented in the following tables. Table 3 shows the values for two important aspects of simulated MRE maps $M_1$, $M_2$ and $M_3$. Table 4 shows the values for the same aspects of the MRE maps $M_1'$, $M_2'$ and $M_3'$, which are the results of applying RSDDM to $M_1$, $M_2$ and $M_3$, respectively.

A quick examination of the results shows that the greatest increase in groups occurs in the $\alpha\beta$ SRE maps. This is not surprising considering that groups in the original $\alpha\beta$ SRE maps tend to contain more virtual fragments than groups in the original $\alpha$ and $\beta$ SRE maps, meaning that there is more potential for improvement in the $\alpha\beta$ SRE maps.

There also tend to be more phantom virtual fragments in the resultant $\alpha\beta$ SRE maps than in the resultant $\alpha$ and $\beta$ SRE maps. This is not surprising either, given the implications of Theorem 18.

There are three properties desireable for the resultant MRE maps:

$(dp_1)$ The partial order of the actual virtual fragments in the resultant MRE maps does not contradict the underlying reality.

<table>
<thead>
<tr>
<th>MRE Map</th>
<th>SRE Map</th>
<th>Fragment Groups</th>
<th>Virtual Fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>$\alpha$</td>
<td>88</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>91</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>$\alpha\beta$</td>
<td>136</td>
<td>164</td>
</tr>
<tr>
<td>$M_2$</td>
<td>$\alpha$</td>
<td>93</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>98</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>$\alpha\beta$</td>
<td>150</td>
<td>187</td>
</tr>
<tr>
<td>$M_3$</td>
<td>$\alpha$</td>
<td>27</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>28</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>$\alpha\beta$</td>
<td>47</td>
<td>59</td>
</tr>
</tbody>
</table>

Table 3
Original MRE maps

<table>
<thead>
<tr>
<th>MRE Map</th>
<th>SRE Map</th>
<th>Fragment Groups</th>
<th>Virtual Fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1'$</td>
<td>$\alpha$</td>
<td>88</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>91</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>$\alpha\beta$</td>
<td>154</td>
<td>173</td>
</tr>
<tr>
<td>$M_2'$</td>
<td>$\alpha$</td>
<td>95</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>101</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>$\alpha\beta$</td>
<td>171</td>
<td>196</td>
</tr>
<tr>
<td>$M_3'$</td>
<td>$\alpha$</td>
<td>29</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>29</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>$\alpha\beta$</td>
<td>54</td>
<td>62</td>
</tr>
</tbody>
</table>

Table 4
New MRE maps
(dp2) The phantom virtual fragments correspond correctly to unregistered fragments in the underlying reality.

(dp3) There is only one virtual fragment per group.

The proof presented in §3.4 guarantees that dp1 is satisfied under ideal conditions. However, MRE maps \(M'_1\), \(M'_2\) and \(M'_3\) were not produced under such ideal conditions, so \(M'_1\), \(M'_2\) and \(M'_3\) are not guaranteed to satisfy dp1. Fortunately, it happens that \(M'_1\), \(M'_2\) and \(M'_3\) do satisfy dp1.

Whether or not dp2 is satisfied depends on the how one defines the correctness of phantom virtual fragments: the difficulty is that there are varying levels of "correctness." For the purpose of this analysis, the following levels of correctness are loosely defined: a phantom virtual fragment pf is considered correct iff (1) the location of pf corresponds to the location of an unregistered genomic fragment uf, (2) pf is within range of uf and (3) ACS(uf) = ACS(pf). A phantom virtual fragment pf is considered overcorrect iff (1) the location of pf corresponds to the location of an unregistered genomic fragment uf, (2) pf is within range of uf, (3) ACS(uf) < ACS(pf) and (4) pf is marked as having an inexact ACS. A phantom virtual fragment pf is considered semicorrect iff (1) the location of pf corresponds roughly to the location of two or more nearby unregistered genomic fragments and (2) pf is within range of the sum of the lengths of the unregistered fragments. A phantom virtual fragment pf is considered wrong iff pf is not correct, overcorrect or semicorrect. Table 5 classifies the phantom virtual fragments in \(M'_1\), \(M'_2\) and \(M'_3\) based on these levels of correctness.

If property dp3 is satisfied, then the exact sequence of the virtual fragments is known. Unfortunately, satisfying property dp3 is usually impossible, due to (1) clusters of unregistered fragments, (2) sequences of consecutive sites of the same enzyme and (3) clone-end incompatibility.

Recall that RSDDM is guaranteed to be correct only if there are no unregistered \(\alpha\) or \(\beta\) genomic fragments in any region and at most one unregistered \(\alpha\beta\) genomic fragment in the gut of any region. Where there are clusters of unregistered genomic fragments that violate this condition, the behavior of RSDDM is somewhat unpredictable. Most of the time, RSDDM simply fails to find any complete and consistent sequences for regions containing such a cluster and the groups in those regions are not refined during the Best MRE Map Phase. Therefore, unless each group contained only one virtual fragment to begin with, dp3 is not satisfied.

<table>
<thead>
<tr>
<th>MRE Map</th>
<th>SRE Map</th>
<th>Correct</th>
<th>Over-Correct</th>
<th>Semi-Correct</th>
<th>Wrong</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M'_1)</td>
<td>(\alpha)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(\beta)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(\alpha\beta)</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>(M'_2)</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(\beta)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(\alpha\beta)</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>(M'_3)</td>
<td>(\alpha)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(\beta)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(\alpha\beta)</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5
Phantom virtual fragment classification
If a stretch of DNA contains many consecutive sites from a single enzyme *enz* (i.e., no sites from the other enzyme are present), then the virtual fragments in the *enz* SRE map that correspond to that stretch cannot be separated by RSDDM. Suppose $g_1, \ldots, g_N$ are consecutive registered genomic fragments created by a set of consecutive $\alpha$ sites. Let $v_1, \ldots, v_N$ be the virtual fragments corresponding to $g_1, \ldots, g_N$ in the $\alpha$ SRE map. Assume that $v_1, \ldots, v_N$ are in the same group. There are $\alpha \beta$ digest genomic fragments that correspond exactly to the genomic fragments $g_1, \ldots, g_N$, because no $\beta$ sites exist in this stretch. When rddm_region is applied to the region containing $v_1, \ldots, v_N$, many complete and consistent sequences will be found for this region. This is because each $v_i$ has a singlet, and any permutation of $\{v_1, \ldots, v_N\}$ forms a complete and consistent sequence. The group containing $v_1, \ldots, v_N$ is not refined during the Best MRE Map Phase, and $dp_3$ is not satisfied.

RSDDM can be applied to MRE maps that are phantom clone-end compatible. However, recall from §4.2 that a group, $g$, containing a clone end that is incompatible with another clone end is never within a processed region. Thus, the virtual fragments in $g$ are never in a complete and consistent sequence, $g$ is not refined during the Best MRE Map Phase, and $dp_3$ is not satisfied.

If $dp_3$ is satisfied for an SRE map, then the number of groups containing virtual fragments is equal to the number of virtual fragments. Thus, one metric for measuring how close an SRE map $m$ is to satisfying $dp_3$ is $(N_{v}(m) - N_{g}(m)) / N_{v}(m)$, where $N_{v}(m)$ is the number of virtual fragments in $m$ and $N_{g}(m)$ is the number of groups containing virtual fragments in $m$. Table 6 gives the value of this metric for $M_1'$, $M_2'$, and $M_3'$, and shows what portion of the metric's value is due to each of the reasons discussed above.

5.2. Time Analysis

In general, the execution time of the entire algorithm seems to be on the order of a small polynomial, specifically quadratic. As evidence for this hypothesis, consider the graph shown in Figure 118 which represents two forms of the execution data. This is the overlay of two plots produced by Mathematica[24]. One is a scatter plot of the total execution time required by the algorithm (as a function of the number of genomic fragments in the double digestion) for 20 different simulated YACS to which the algorithm has been applied. The second is a continuous plot of the function produced by the Fit operator of Mathematica, asked to produce the best least-square fit (of the scatter plot data) to a 2nd degree polynomial. The rightmost point in the plot corresponds to a 1400kb YAC containing 726 genomic $\alpha \beta$ fragments. The next point in the plot corresponds to a 1200kb YAC containing 577 genomic $\alpha \beta$ fragments. (Note that this

<table>
<thead>
<tr>
<th>MRE Map</th>
<th>SRE Map</th>
<th>Clusters</th>
<th>Consecutive</th>
<th>Incompatibility</th>
<th>Other/Unknown</th>
<th>$(N_{v}-N_{g})/N_{v}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1'$</td>
<td>$\alpha$</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>10/98</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>11/102</td>
</tr>
<tr>
<td></td>
<td>$\alpha \beta$</td>
<td>4</td>
<td>13</td>
<td>0</td>
<td>2</td>
<td>19/173</td>
</tr>
<tr>
<td>$M_2'$</td>
<td>$\alpha$</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>9/104</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>8/109</td>
</tr>
<tr>
<td></td>
<td>$\alpha \beta$</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>25/196</td>
</tr>
<tr>
<td>$M_3'$</td>
<td>$\alpha$</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4/33</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2/31</td>
</tr>
<tr>
<td></td>
<td>$\alpha \beta$</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>8/62</td>
</tr>
</tbody>
</table>

Table 6

Reasons for groups with more than one virtual fragment
Figure 118: Total execution time

point seems to be an outlier. Such data points should be expected in the results of any set of experiments based on random occurrences.) The next point in the plot corresponds to a 1000kb YAC containing 494 genomic $\alpha\beta$ fragments. The next two points in the plot correspond to an 800kb YAC containing 399 genomic $\alpha\beta$ fragments. (Both of these correspond to the same simulated YAC. One map corresponds to the "best" MRE map that can be produced from the simulated data; the other is a map obtained by a human mapping the data with our MRE mapping software.) The rest of the points correspond to random selections of small MRE maps collected from various sources. Since this algorithm is intended to be applied to YACs in approximately the 800-1000kb range, presenting data in this range seems appropriate.

Note that the quadratic produced by Fit seems to fit the data relatively well, except for the last two points. If the point corresponding to the 1200kb YAC is removed, the fit is much better. The equation for the least-square fit (produced by Fit) is:

$$-295.538 + 6.08282x + 0.00695225x^2$$

Other experiments (not shown here) were performed to fit these data to an exponential; the resulting "best fit function" obtained did not conform well to these data.

In order to understand the computational complexity of the different components of the algorithm, a separate analysis will be done for each of the four phases: Phase I - Region Sequence Phase, Phase II - RDDM Phase, Phase III - Phantom Fragment Equivalence Phase, and Phase IV - Best MRE Map Phase. A preview of the analysis results is presented here, before the detailed analyses. The execution time of Phase I is quadratic in the number of clones present. This is because, during the region building activity, each left
clone end must be considered with each right clone end in order to determine whether or not an appropriate region is defined by them. Phase I is the dominant time-consumer of all four phases, usually taking more than 90% of the total time.

The execution time of Phase II seems to be just slightly sublinear in the number of fragments present. A convincing argument can be given for this observed performance, although, in general, the theoretical upper bound is still exponential in nature.

The theoretical upper bounds of Phases III and IV are also exponential. Phase III seems to have slightly sublinear observed performance, and Phase IV's seems to be slightly quadratic.

5.3. Phase I - Region Sequence Phase

The execution-time scatter plot and best-fit quadratic for Phase I are shown in Figure 119. The curve for Phase I looks very similar to the overall curve. Since in most of the test executions, roughly 95% of the runtime is spent in Phase I, this is not surprising. The equation for the best least-square fit (produced by Fit) is:

\[-288.116 + 5.63597x + 0.00692261x^2\]

If the outlier corresponding to the 1200kb YAC is removed, the corresponding best least-square fit becomes much flatter.

![Figure 119: Phase I execution time](image-url)
In Phase I, all incompatible left clone-end/right clone-end pairs in the given MRE map are computed. In order to do this, the positional relationship of each left clone end and right clone end must be calculated. A brute force method that compares each left clone end with each right clone end has a worst-case runtime of $O(N_c^2)$. ($N_c$ is the maximum number of clones in an SRE map of the MRE map.) The maximum depth of an SRE map is bounded by the number of clones available for incorporation, though in practice, the bound is significantly lower. The number of clones in an MRE map is typically proportional to the number of virtual fragments in the MRE map, so thus, it can be said that the runtime of Phase I is quadratic with respect to the number of virtual fragments in the MRE map. If this method is actually implemented, the actual runtime is indeed the worst-case runtime.

In the implementation of RSDDM, therefore, a slightly more sophisticated method is used. This method scans each SRE map from left to right so that many of the compatible clone-end pairs are never examined. Although the worst-case runtime of this method is no better than the brute-force method, the runtime for typical cases is usually significantly less.

5.4. Phase II - RDDM Phase

The execution-time scatter plot and best fit quadratic are shown in Figure 120. Note how flat the curve seems to be: just slightly sublinear. The equation for the best least-square fit (produced by Fil) is:

$$0.73651 + 0.193467x - 0.000134548x^2$$

![Figure 120: Phase II execution time](image)
If the two outliers corresponding to the 800kb YAC are removed, this curve becomes even flatter, *i.e.*, closer to linear than sublinear. These data suggest that the execution time for this phase might be roughly linear in the number of fragments present in the entire map. Consider the following basic logic for why this is feasible (there are several flaws in this logic which will be addressed at the end of the discussion).

In general, the execution time of the RDDM phase is exponential as a function of the number of fragments it processes. In this implementation, these fragments come from the gut of a region. If the number of fragments in the gut of any region can be kept uniformly bounded from above, then the execution time for processing the fragments in the gut of the region can likewise be uniformly bounded from above. More specifically, if \( n < K \), for some constant \( K \), then \( e^n < e^K = C \), for another constant \( C \). If the number of guts processed over the entire genome is \( G \), then the overall execution time can be bounded by \( G^C \). Since the number of guts (\( G \)) over the genome is linear in the number of fragments present (each gut must contain at least one fragment), the execution time can be bounded by a function which is linear in the number of fragments present.

In fact, the number of fragments in the gut of a region can, for the most part, be bounded. A reasonable upper bound is the number of fragments in the largest clone used in constructing the map. No group in the MRE map produced will contain any more fragments than the number present in the largest clone. Since the left and right clone ends of each specific clone are used as candidates for delimiting regions during the region sequence phase, no basic region will ever be larger than the largest clone. In general, no gut of a non-basic region will be larger than the largest clone either. The reason for this is that for a gut to be larger than a clone, it would have to contain both the left and right clone end of some specific clone. However, somewhere in the region delimited by these two clone ends, there must exist a basic region—*e.g.*, the region delimited by the two clone ends themselves. Thus, basic regions remain relatively small and the gut of an ever-enlarging non-basic region also remains relatively small, thus supplying a uniform upper bound for the number of fragments processed by the a single invocation of RDM.

There are two major flaws in the basic logic presented above, but the situations in which these flaws become evident are relatively infrequent in practice. First, it is actually possible for a gut to be larger than the largest clone. This, for instance, occurs when there is no overlap between the \( \alpha \) and \( \beta \) digestions, as depicted in Figure 27(C). In such a case, the formation of a basic region is deferred to include surrounding portions of the genome until fragment overlap between the \( \alpha \) and \( \beta \) digestions occurs.

Second, the fragments in the gut of a region are processed for each seed extracted from an adjacent left and/or right helper-region. Of course, each seed corresponds to a currently compatible sequence of fragments which could occur in that helper-region, for which any fragment sequence in the gut being processed will simply be an extension. In general, it may be possible to bound the number of seeds produced by a basic region (since the number of fragments in a basic region can effectively be bounded), but the number of seeds can grow exponentially as the guts of larger and larger regions are processed (*i.e.*, if sequences produced for the gut do not exclude previously compatible sequences produced by the helper-regions). Such an exponential growth of the number of compatible sequences would, of course, produce a corresponding exponential growth in the execution time. However, in practice, the number of sequences does not seem to grow exponentially. The incremental process of attempting to extend sequences from helper-regions into the gut of a new region tends to prune a significant number of the sequences which were previously compatible in the helper-region. These sequences, while compatible outside of the surrounding context, become incompatible when put into the context of the guts of surrounding regions. Thus, in practice, the number of compatible sequences does not tend to grow exponentially as the genome increases in size.
5.5. Phase III - Phantom Fragment Equivalence Phase

The execution-time scatter plot and best fit quadratic for Phase III are shown in Figure 121. The curve for Phase III appears to be sublinear. The equation for the best least-square fit (produced by Fit) is:

\[-0.00715673 + 0.0166507x - 0.0000156291x^2\]

Again, if the two outliers corresponding to the 800kb YAC are removed, this curve becomes even flatter, i.e., closer to linear than sublinear.

The runtime of Phase III depends primarily on two factors. The first is the time required to determine whether a loose phantom set is member-equivalent. The second is the number and size of the loose phantom sets after the completion of Phase II.

The time required for checking member-equivalence is now examined. Suppose $S$ is a loose phantom set. Conditions $me_1$ and $me_2$ can be checked in $O(|S|)$ time. However, checking condition $me_3$ is more time consuming. Let $K$ be the size of the largest element in any ACS path of all the virtual fragments in $S$. A rough analysis indicates that $O(|S|K^2)$ time is required to check $me_3$. Suppose it is assumed that the depth of each SRE map in the MRE map is bounded by some constant. Then $K$ is bounded by that constant, and thus it takes $O(|S|)$ time to check $me_3$. Thus, the time required to check for member-equivalence of $S$ is $O(|S|)$.

![Figure 121: Phase III execution time](image)
Now the number and size of loose phantom sets is examined. Let $N_R$ be the number of regions in the MRE map. Let $N_S$ be the number of compound virtual fragment sequences in the region which contains the most sequences. Then there are at most $3N_R$ loose phantom sets in the MRE map. The largest any of those loose phantom sets could be is $N_S$. Recall that it takes $O(|S|)$ time to determine the member-equivalence of a loose phantom set $S$. Thus, the worst-case runtime for Phase III is $O(N_R N_S)$. Theoretically, $N_S$ can grow exponentially with respect to the number of virtual fragments in the MRE map. However, for the cases to which RSDDM has been applied, $N_S$ grows at a much lower rate.

5.6. Phase IV - Best MRE Map Phase

The execution-time scatter plot and best fit curve for Phase IV are shown in Figure 122. The curve appears to be somewhere between linear and quadratic. The equation for the best least-square fit (produced by Fit) is:

$$-3.34272 + 0.126887x + 0.000322069x^2$$

Suppose that a group containing $N_1$ virtual fragments lies in a region containing $N_2$ compound virtual fragment sequences at the end of Phase III. The worst-case runtime for finding the best refinement (i.e., subgrouping of the fragments) of that particular group is $O(N_1^2 N_2^2)$.

![Figure 122: Phase IV execution time](image-url)
Let $N_G$ be the number of groups in an SRE map $s$. Let $M_1$ be the maximum number of virtual fragments in a group of $s$. Let $M_2$ be the maximum number of sequences for a region of $s$ after Phase III. Then the worst-case runtime for processing $s$ in Phase IV is $O(N_G M_1^2 M_2^2)$. The following characteristics of the SRE maps in a high quality MRE map prevent the runtime of Phase IV from actually being this bad.

1. $N_G$ is the number of virtual fragments in the SRE map.
2. $M_1$ is usually relatively small, even in large SRE maps.
3. $M_2$ is usually relatively small, even in large SRE maps. (See §5.4.)
4. $N_G$ and $M_1$ vary inversely.

These characteristics result in a typical runtime for Phase IV that grows quadratically with respect to the number of virtual fragments in the MRE map.

### 5.7. Is This Magic?

All algorithms for solving the double-digest mapping problem are known, both in theory and in practice, to be exponential in nature. The technique presented here seems to be quadratic in nature. (In theory, it is still exponential, but empirically it seems to be polynomial.) Is this MRE mapping, divide-and-conquer approach, a magical bridge that proves that NP-hard problems are really polynomial in nature? Unfortunately, the answer is "no!"

It turns out that the MRE mapping problem is NP-hard, and it is, therefore, also exponential in nature. Much of the work that would have gone into solving the DDM problem for a very large unstructured genome has simply been transferred to the process of creating the MRE map, i.e., creating a structured genome to which the DDM algorithm can be applied in a piecemeal manner. Thus, the apparent magic here is similar to that of all sleight-of-hand, diverting the attention from where the real activity is occurring.

Another caveat has to do with the seemingly polynomial nature of the actual execution times presented in this section. It should be made clear that given any exponential curve, a small enough region extracted from it will always appear to be polynomial in nature when removed from its global context. Thus, while the nature of this approach seems to be well behaved in the region analyzed, it is possible that underlying exponential behavior simply has not yet become apparent. However, since this is the region in which this approach is likely to be applied, the claim of polynomial performance seems justified.

### 6. Conclusion

In this paper, two primary algorithms were presented: ISDDM and RSDDM.

ISDDM takes an MRE map composed of three SRE maps (an $\alpha$ map, a $\beta$ map, and an $\alpha \beta$ map) and returns a new version of that MRE map. The new version may contain more groups than the original, and thus it may encode a more specific partial order of the fragments than the original. ISDDM accomplishes this by using a technique that is similar to DDM algorithms used in the past upon single clone data, except that this technique has been adapted to operate on small portions of an MRE map. This technique is applied repeatedly to different portions of the MRE map. This allows ISDDM to run in a reasonable time on MRE maps of considerable size. Unfortunately, ISDDM makes some unreasonable assumptions about the conditions that existed for the creation of the original MRE map.
RSDDM is similar to ISDDM, except that it is able to operate on MRE maps created under more realistic conditions. In addition to creating more groups, RSDDM can make hypotheses about the lengths and locations of genomic fragments that are too small to be detected by the agarose gel electrophoresis process. Although there is no absolute guarantee that these hypotheses are correct, this ability should prove to be a useful feature, especially if coupled with a mechanism that can verify these hypotheses with data from further biological experiments.

Improvements in the method of hypothesizing the lengths and locations of large unregistered genomic fragments, and decreasing the number of incorrect hypotheses about unregistered genomic fragments would make RSDDM even more useful.
APPENDIX A

Description of Functions not Defined by Pseudocode

This appendix describes the functionality of operators used in the pseudocode as primitives, for which no pseudocode was given.

LIST
ce_sort_by_position(s,m,dir)
  SET s;
  MREMAP m;
  DIRECTION dir;

This function returns a list of the clone ends in the set s, sorted on their positions in the MRE map m and the direction dir. If dir = LEFT, then the leftmost clones appear first in the list. If dir = RIGHT, then the rightmost clones appear first in the list.

SET
ces_of(o,dir)
  GROUP o; or SREMAP o;
  DIRECTION dir;

This function returns the set of clone ends with direction dir in the group or SRE map o.

ENZYMEE
enzyme_of(vf)
  VIRTUAL_FRAGMENT vf;

This function returns the enzyme from which the virtual fragment vf originates (i.e., $\alpha$, $\beta$ or $\alpha \beta$).

ENZYMEE
enzyme_of_sole_parent(q,vf)
  CSEQUENCE q;
  VIRTUAL_FRAGMENT vf;

This function returns the enzyme of the parent of the virtual fragment vf in the compound sequence q (i.e., $\alpha$ or $\beta$). It is assumed that vf has only one parent in q.

CSEQUENCE
eempty_compound_seq()

This function returns an empty compound virtual fragment sequence.

TUPLE
empty_tuple(n)
  INTEGER n;

This function returns a tuple of size n with all NULL elements.

LENGTH
frag_length_of(vf)
    VIRTUAL_FRAGMENT vf;

This function returns the length of the virtual fragment vf.

SET
groups_between_ces(s,c1,c2)
    SREMAP s;
    CLONE_END c1,c2;

This function returns the set of groups in the SRE map s that are between the groups containing clone ends c1 and c2.

VOID
group_add_objects(g,s)
    GROUP g;
    SET s;

This function adds the objects in the set s to the group g.

GROUP
group_containing(s,o)
    SREMAP s;
    VIRTUAL_FRAGMENT o; or CLONE_END o;

This function returns the group in the SRE map s that contains the virtual fragment (or clone end) o.

GROUP
group_create(s)
    SET s;

This function returns a group containing the objects in the set s.

INTEGER
group_num_of(s,g)
    SREMAP s;
    GROUP g;

This function returns an integer i where g is the i-th group in the SRE map s.

BOOLEAN
is_left_clone_end(o)
    any type o;

This function returns TRUE if the object o is a left clone end. Otherwise, FALSE is returned.

BOOLEAN
is_right_clone_end(o)
    any type o;

This function returns TRUE if the object o is a right clone end. Otherwise, FALSE is returned.
BOOLEAN
is_left_of(s,c1,c2)
  SREMAP  s;
  CLONE_END  c1,c2;

This function returns TRUE if the group containing clone end c1 is to the left of the group containing the clone end c2 in the SRE map s.

BOOLEAN
is_right_of(s,c1,c2)
  SREMAP  s;
  CLONE_END  c1,c2;

This function returns TRUE if the group containing clone end c1 is to the right of the group containing the clone end c2 in the SRE map s.

LENGTH
length_of_vfrags_between_ces(s,c1,c2)
  SREMAP  s;
  CLONE_END  c1,c2;

This function returns the sum of the lengths of the virtual fragments in the SRE map s that are in groups between the groups containing the clone ends c1 and c2.

LIST
list_left_elements_of(x)
  LIST  x;

The list x must contain only pairs. This function returns a list containing the left element of each pair in x.

LIST
list_right_elements_of(x)
  LIST  x;

The list x must contain only pairs. This function returns a list containing the right element of each pair in x.

LIST
list_singletonize_elements(x)
  LIST  x;

This function returns a list whose i\textsuperscript{th} element is [\phi], where \phi is the i\textsuperscript{th} element of x.

GROUP
nth_group_of(s,n)
  SREMAP  s;
  INTEGER  n;

This function returns the n\textsuperscript{th} group (from the left) of the SRE map s.

INTEGER
num_groups_of(s)
  SREMAP  s;
This function returns the number of groups in the SRE map $S$.

```plaintext
INTEGER
num_vfrag_between_ces(s,c1,c2)
  SREMAP s;
  CLONE_END c1,c2;
```

This function returns the number of virtual fragments in the SRE map $S$ that are in groups between the groups containing the clone ends $c_1$ and $c_2$.

```plaintext
ENZYME
other_enzyme(enz)
  ENZYME enz;
```

This function returns $\alpha$ if $\text{enz} = \beta$, and it returns $\beta$ if $\text{enz} = \alpha$.

```plaintext
LIST
region_sort_by_size(m,s)
  MREMAP m;
  SET s;
```

This function returns a list of the regions in the set $S$, sorted based upon their size, as defined in §3.1.

```plaintext
VOID
remove_helper_fragments(q)
  CSEQUENCE q;
```

This procedure removes helper-fragments from the compound virtual fragment sequence $q$.

```plaintext
BOOLEAN
seq_is_complete(q,x,enz)
  CSEQUENCE q;
  LIST vfs;
  ENZYME enz;
```

This function returns TRUE if the $\text{enz}$ digest sequence of the compound virtual fragment sequence $q$ is complete with respect to the list of virtual fragments $vfs$ (i.e., it contains all members of $vfs$). Otherwise, FALSE is returned.

```plaintext
BOOLEAN
seq_is_empty(q)
  CSEQUENCE q;
```

This function returns TRUE if the compound virtual fragment sequence $q$ is empty (i.e., contains no virtual fragments.) Otherwise, FALSE is returned.

```plaintext
SEED
shape_of_end(q,dir)
  CSEQUENCE q;
  DIRECTION dir;
```

This function returns a seed which describes the shape of the $\text{dir}$ end of the compound virtual fragment
sequence q.

SREMAP
sremap_create(x)
  LIST   x;

This function returns an SRE map composed of the groups in the list x, in the order specified by x.

HELPER_FRAGMENT
helper_fragment_create(s)
  SEED   s;

This function returns a helper-fragment that corresponds to the seed s.

VERTEX
vertex_create(o)
  any type o;

This function returns a vertex of a graph which contains the object o as its data.

SET
vfrags_of(g)
  GROUP   g;

This function returns the set of virtual fragments in the group g.

BOOLEAN
vfrag_is_in_seq(q,vf)
  CSEQUENCE   q;
  VIRTUAL_FRAGMENT   vf;

This function returns TRUE if the virtual fragment vf is in the compound virtual fragment sequence q. Otherwise, it returns FALSE.

BOOLEAN
within_range(len1,len2)
  LENGTH   len1,len2;

This function returns TRUE if the lengths len1 and len2 are the same, taking measurement error into account. Otherwise, it returns FALSE.
APPENDIX B

Notational Conventions in the Pseudocode

This appendix defines the symbolic notation used in the body of the report.

Ø the empty set

|s| the cardinality of a set s

[] the empty list

[0,1,0,2,...,0,N] a list containing N objects

|x| the cardinality of a list x

x[i] the ith element of a list x

x[i,...,j] the sublist of a list x formed by the ith through jth elements

x[i,...] the sublist of a list x formed by the ith through last elements

[] the list concatenation operator

(0,1,0,2,...,0,N) a tuple containing N objects

t[i] the ith element of a tuple t

| f_1=v_1, f_2=v_2, ..., f_N=v_N | a record-like structure with N fields where f_i is the ith field name and v_i is the value of the ith field.
References


