Combinatorial Algorithms on Crossover

Crossover distance

Problem: Given two genomes G and G' what is the minimal number of crossover mutations that transforms G into G'?

1. How the crossover is defined: uniform or arbitrar.

2. How the chromosomes which form the genomes are given: different segments (markers) or not.

Formal definitions

For each string $x \in V^+$, whose length is denoted by $|x|$, $x[i,j]$ delivers the substring of x that starts at position i and ends at position j in x, $1 \leq i \leq j \leq |x|$.

For two strings x, y over an alphabet V and two integers $1 \leq i < |x|, 1 \leq j < |y|$, we define the crossover operation

$(x,y) \vdash_{(i,j)} (z_1, z_2)$ iff $x = tu, y = vw, z_1 = tw, z_2 = vu,$ and $|t| = i, |v| = j$.

Formally, the crossover operation $\vdash_{(i,j)}$ is said to be *uniform* iff $i = j$, so that we shall simply write $\vdash_i$.

We extend the (uniform) operation to a finite set of strings $A \subseteq V^+$ by $(U)CO(A) = \bigcup_{x,y \in A} \{z, w | (x,y) \vdash (z,w)\}$.

Uniform crossover and unique markers
(J. Kececioglu, R. Ravi 1995)

Assumptions:
1. All chromosomes (words) in both genomes are of the same length k.
2. Each marker (symbol) appears at most once in a chromosome and in only one.
3. If G has n chromosomes, then G' must have n chromosomes as well.

Important note: If a symbol appears on the position i in a word in G, then it will appears on the same position in a word of G'.

1. We label the words in G' in some way from 1 to n.
2. Associate with each set G, G' a matrix as follows:

- each column in the matrix represents a word

- each symbol from a word is represented by the label of the unique word of G' in which it occurs. 3. Each row in the matrix associated with G is a permutation of the first n natural numbers.

4. Each row in the matrix associated with G' is the identical permutation ε_n.

Example: $G = \{a_2a_7a_9a_4, a_5a_1a_{12}a_8, a_{10}a_3a_6a_{11}\}$
$G' = \{a_{10}a_1a_9a_8, a_5a_7a_6a_4, a_2a_3a_{12}a_{11}\}$

$$M_G = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix} \quad M_{G'} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

Problem: Select two columns and a natural $l \leq n-1$ and interchange the elements of the first l rows.

How to compute the minimal number of these selections? **Greedy strategy**

Let (i, j, l): the columns i and j interchange each other the entries of the first l rows. A solution is a sequence

$(i_1, j_1, l_1), (i_2, j_2, l_2), \ldots (i_p, j_p, l_p)$

Find the minimal p.

A solution $(i_1, j_1, l_1), (i_2, j_2, l_2), \ldots (i_p, j_p, l_p)$ is "bottom-up if there are no $1 \leq s < q \leq n - 1$ such that $l_q > l_s$.

Lemma 1 *Any instance of the problem has a solution which is bottom-up.*

Proof. Let

$$(i_t, j_t, l_t),\ (i_{t+1}, j_{t+1}, l_{t+1}),$$

cu $l_t < l_{t+1}$. W.l.o.g we assume that $i_t \neq i_{t+1}$ and $j_t = j_{t+1}$.

Replace this pair by

$$(i_{t+1}, j_{t+1}, l_{t+1}),\ (i_t, i_{t+1}, l_t),$$

□

A bottom-up sequence is *locally optimal* if the number of transformations applied to the current row in order to transform it into the identical permutation is minimal.

Lemma 2 *A bottom-up locally optimal is totally optimal.*

Proof. Let us consider a part of a bottom-up sequence when one starts to "sort the row $i+1$. Let π be the current state of the row $i+1$ and λ_i the state of the row. After sorting the row $i+1$ the state of the row i is

$$\lambda_i \circ \pi^{-1}.$$

□

Given a permutation π, what is the minimal number m of transpositions $\tau_1, \tau_2, \ldots, \tau_m$ such that

$$\pi \circ \tau_1 \circ \tau_2 \circ \ldots \circ \tau_m = \varepsilon_n$$

Solution: Cayley (1849)

Let $\Psi(\pi)$ its number of cycles. A cycle of π is a sequence of numbers between 1 and n such that

$$i_0 = i, i_1, i_2, \ldots, i_t, \ t \geq 0,$$

with

$$\pi(i_j) = i_{j+1}, \ 0 \leq t-1, \ \pi(i_t) = i.$$

Lemma 3 (Cayley) *The minimal number of transpositions for sorting* π *is* $n - \Psi(\pi)$.

procedure Sort_Crossover_uniform(A,k,n);

Let $\lambda_1, \lambda_2, \ldots, \lambda_k$ *the rows of A*
$d := 0$; $\pi := \varepsilon_n$;
for $i := k$ **downto** 1 **do**
    $\pi := \lambda_i \circ \pi^{-1}$;
    $d := d + n - \Psi(\pi)$;
endfor;
end.

Theorem 1 *The uniform crossover distance between G and G' can be computed in time and memory $O(kn)$.*

Uniform crossover without unique markers
(C. Martin-Vide, V. Mitrana 2004)

Let A be a finite set of strings such that each string of A has arbitrarily many available copies. Define iteratively

$$CO_0(A) = A, \quad CO_{k+1}(A) = CO_k(A) \cup CO(CO_k(A))$$

$$CO_*(A) = \bigcup_{k \geq 0} CO_k(A).$$

A *crossover sequence* in $CO_*(A)$ is a sequence $S = s_1, s_2, \ldots, s_n$, where for each $1 \leq i \leq n$ $s_i = (x_i, y_i) \vdash_{(k_i, p_i)} (u_i, v_i)$, for some $x_i, y_i, u_i, v_i \in CO_*(A)$ and $1 \leq k_i < |x_i|$, $1 \leq p_i < |y_i|$.

Given a crossover sequence S as above and $x \in TO_*(A)$ we define

$$P_i(S,x) = card\{j \leq i | x = x_j \text{ or } x = y_j\} + card\{j \leq i | x_j = y_j = x\},$$

$$F_i(S,x) = \begin{cases} card\{j \leq i | u_j = x_j \text{ or } v_j = y_j\} + card\{j \leq i | u_j = v_j = x\}, \text{if } x \notin A, \\ \infty, \text{ otherwise.} \end{cases}$$

The *length* of a crossover sequence $S = s_1, s_2, \ldots, s_n$ is denoted by $lg(S)$ and equals n.

A crossover sequence S as above is *contiguous* iff the following two conditions are satisfied:

(i) $x_1, y_1 \in A$,

(ii) $F_{i-1}(S, x_i) > P_{i-1}(S, x_i)$, and $F_{i-1}(S, y_i) > P_{i-1}(S, y_i)$, for all $1 \leq i \leq n$.

By CTS we mean a contiguous crossover sequence.

Let B be a finite subset of $CO_*(A)$; a CTS S as above is B-producing if $F_n(S,z) > P_n(S,z)$ for all $z \in B$.

$$CD(A,B) = \min\{lg(S) | S \text{ is a } B- \text{ producing } CTS \text{ in } CO_*(A)\}.$$

Example 1 *Let us consider the initial set* $A = \{x_1, x_2, x_3, x_4\}$ *with* $x_1 = abcbad$, $x_2 = bbabd$, $x_3 = accbabd$, $x_4 = aaab$, *and*

$z_1 = bbcbad$ $\quad z_2 = ababd$ $\quad z_3 = ababad$ $\quad z_4 = bbcbd$
$z_5 = abbababd$ $\quad z_6 = aabad$ $\quad z_7 = abababd$ $\quad z_8 = bbd$
$z_9 = bbbd$ $\quad z_{10} = bbabad$ $\quad z_{11} = bbbabad$ $\quad z_{12} = bbababd$
$z_{13} = bababd$ $\quad z_{14} = accbd$ $\quad z_{15} = bbccbabd$ $\quad z_{16} = aababd$
$z_{17} = abcccbabd$ $\quad z_{18} = abad$

We provide below a B-producing CTS, $B = \{z_4, z_6, z_8, z_{11}, z_{15}, z_{16}, z_{18}\}$.

$(x_1, x_2) \vdash_{(2,2)} (z_2, z_1)$, $(z_1, z_2) \vdash_{(4,4)} (z_4, z_3)$, $(x_1, x_2) \vdash_{(2,2)} (z_2, z_1)$,
$(z_2, x_2) \vdash_{(4,2)} (z_7, z_8)$, $(z_3, z_7) \vdash_{(2,1)} (z_5, z_6)$, $(x_2, x_3) \vdash_{(3,3)} (z_{12}, z_{14})$,
$(z_8, z_{12}) \vdash_{(2,5)} (z_9, z_{10})$, $(x_2, x_3) \vdash_{(3,3)} (z_{12}, z_{14})$, $(x_2, x_3) \vdash_{(3,3)} (z_{12}, z_{14})$,
$(z_{12}, z_{10}) \vdash_{(2,1)} (z_{11}, z_{13})$, $(z_{12}, x_3) \vdash_{(2,1)} (z_{15}, z_{16})$,
$(x_1, x_3) \vdash_{(3,1)} (z_{17}, z_{18})$.

Singleton Target Sets: Greedy strategy

We may assume that the initial set of strings contains only strings of the same length, that is the length of the target string. (*Simple argument*)

Let $A = \{x_1, x_2, \ldots, x_n\}$ and z be an arbitrary string of length k

$$MaxSubLen(A, z, p) = \max\{q |\ \exists\ 1 \leq i \leq n \text{ such that } x_i[p, p+q-1] = z[p, p+q-1]\}.$$

Let $z \in TO_*(A)$; define iteratively the set $H(A, z)$ of intervals of natural numbers as follows:

1. $H(A, z) = \{[1, MaxSubLen(A, z, 1)]\}$;

2. Take the interval $[i, j]$ having the largest j; if $j = k$, then stop, otherwise put into $H(A, z)$ the new interval $[j+1, j+MaxSubLen(A, z, j+1)]$.

Note that we allow intervals of the form $[i, i]$ for some i to be in $H(A, z)$; moreover, for each $1 \leq i \leq k$ there are $1 \leq p \leq q \leq k$ (possibly the same) such that $i \in [p, q] \in H(A, z)$.

Lemma 4 *Let S be a z-producing CTS in* $CO_*(A)$. *Then,*

$$lg(S) \geq card(H(A, z)) - 1.$$

Proof. Induction on the length k of z. Let us consider a CTS $S = s_1, s_2, \ldots, s_q$ in $CO_*(A)$ producing z. Moreover, we may assume that $s_i = (x_i, y_i) \vdash_{p_i} (u_i, v_i)$, $1 \leq i \leq q$, and z has been obtained in S at the last step, that is either $u_q = z$ or $v_q = z$. Let

$$\begin{aligned} A' &= \{x[MaxSubLen(A, z, 1) + 1, k] | x \in A\}, \\ z' &= z[MaxSubLen(A, z, 1) + 1, k]. \end{aligned}$$

For simplicity denote $r = MaxSubLen(A, z, 1)$. Clearly, $H(A', z') = \{[i-r, j-r] | [i, j] \in H(A, z) \backslash \{[1, r]\}\}$, hence $card(H(A', z')) = card(H(A, z)) - 1$. Starting from S we construct a CTS in $CO_*(A')$, producing z' $S' = s'_1, s'_2, \ldots s'_m$ in the way indicated by the following procedure:

```
Procedure Construct_CTS(S,r);
begin
```
$m := 0$;
```
for i := 1 to q begin
     if  (p_i > r)  then
```
$m := m+1;\ s'_m = (x_i[r+1,k], y_i[r+1,k]) \vdash_{p_i - r} (u_i[r+1,k], v_i[r+1,k])$;
```
     endif;
endfor;
end.
```

Claim 1: *S' is a CTS.*

Claim 2: *S' is z'-producing.*

But there exists at least one i such that $p_i \leq r$, it follows that $m \leq q-1$. By the induction hypothesis, $m \geq card(H(A', z')) - 1$, and the proof is complete. □

Theorem 2 *Let z be a string of length k and A be a set of cardinality n. There is an exact algorithm that computes CD(A,z) in O(kn) time and O(kn) space.*

Proof.

```
Procedure Uniform_translocation_CTS_construction(A,z);
begin
p := MaxSubLen(A, z, 1); let x be a string in A with x[1, p] = z[1, p];
m := 0;
while  p < k  begin
    r := MaxSubLen(A, z, p + 1);
     if  r = 0  then THE STRING  z CANNOT BE OBTAINED
FROM  A;  stop
       else
         let y be a string in A with y[p + 1, p + r] = z[p + 1, p + r];
         m := m + 1;  s_m = (x, y) ⊢_p (u, v)};
         p := p + r;  x := u;
     endif
endwhile;
end.
```

□

Arbitrary Target Sets

Let A be a finite set of strings and $z \in CO_*(A)$; denote by

$$
\begin{aligned}
MaxPrefLen(A,z) &= \begin{cases} |z|, \text{ iff } z \in A, \\ \max(\{q | q < |z|, \text{ there exists } x \in A, |x| > q, \\ \quad \text{so that } x[1,q] = z[1,q]\} \cup \{0\}), \end{cases} \\
MaxSufLen(A,z) &= \max(\{q | \text{ there exists } x \in A, |x| \geq |z|, \\
&\quad \text{so that } x[|x|-q+1,|x|] = z[|z|-q+1,|z|]\} \\
&\quad \cup \{0\}), \\
ArbMaxSubLen(A,z,p) &= \max(\{q | \text{ there exists } x \in A \text{ and } |x| \geq p+q \\
&\quad \text{such that } x[p,p+q-1] = z[p,p+q-1]\} \\
&\quad \cup \{0\}).
\end{aligned}
$$

We define iteratively the set $ArbH(A, z)$ of intervals of natural numbers as follows, provided that all parameters defined above are nonzero:

1. $ArbH(A, z) = \{[1, MaxPrefLen(A, z)]\}$;

2. Take the interval $[i, j]$ having the largest j; if $j = |z|$, then stop. If $j < |z| - MaxSufLen(A, z)$, then put the new interval $[j + 1, j + ArbMaxSubLen(A, z, j + 1)]$ into $ArbH(A, z)$; otherwise put $[j + 1, |z|]$ into $ArbH(A, z)$.

Theorem 3 *1. Let A be a finite set of strings and B be a finite subset of* $TO_*(A)$. *Then* $\frac{\sum_{z\in B}(card(ArbH(A,z))-1)}{2} \leq TD(A,B) \leq \sum_{z\in B}(card(ArbH(A,z))-1)$.

2. There exist A *and* $B \subseteq TO_*(A)$ *such that* $TD(A,B) = \frac{\sum_{z\in B}(card(ArbH(A,z))-1)}{2}$.

3. There exist A *and* $B \subseteq TO_*(A)$ *such that* $TD(A,B) = \sum_{z\in B}(card(ArbH(A,z))-1)$.

Proof. 1. We shall prove the first assertion by induction on the length of the longest string in B, say k. The non-trivial relation is

$$\frac{\sum_{z \in B}(card(ArbH(A,z))-1)}{2} \leq TD(A,B). \qquad (*)$$

If $k = 1$, the relation $(*)$ is satisfied. Assume that the relation $(*)$ holds for any two finite sets X and Y, $Y \subseteq TO_*(X)$, all strings in Y being shorter than k. Assume that $B \setminus A = \{z_1, z_2, \ldots, z_m\}$ and let $S = s_1, s_2, \ldots, s_q$, $s_i = (x_i, y_i) \vdash_{p_i} (u_i, v_i)$, $1 \leq i \leq q$, be a $B \setminus A$-producing CTS in $TO_*(A)$. Note that at least one string in $B \setminus A$ should exist, otherwise the relation $(*)$ being trivially fulfilled.

Consider m new symbols $a_1, a_2, \ldots, a_m$ and construct the sets: $A' = \{x[1,r]a_i x[r+2,|x|]|x \in A, 1 \leq i \leq m\}$, $B' = \{z_i[1,r]a_i z_i[r+2,|z_i|]|1 \leq i \leq m\}$,, where $r = \min\{p_i|1 \leq i \leq q\}$. One can construct a B'-producing CTS in $TO_*(A')$ of the same length of S, say S' by applying a procedure $Convert$ illustrated by the next example

$B = \{abacdb,\ aabccb,\ bbaadc\},\ A = \{abbccb, aaaadb, bbbcdc\}.$

The $CTS\ S$ is

$(abbccb, aaaadb) \vdash_2 (abaadb, aabccb), (abbccb, abaadb) \vdash_3 (abbadb, abaccb),$
$(bbbcdc, abaccb) \vdash_2 (bbaccb, abbcdc), (bbaccb, aaaadb) \vdash_3 (bbaadb, aaaccb),$
$(bbaadb, bbbcdc) \vdash_5 (bbaadc, bbbcdb), (abaadb, aaaccb) \vdash_2 (abaccb, aaaadb),$
$(abaccb, aaaadb) \vdash_4 (abacdb, aaaacb).$

The procedure $Convert$ runs for $r = 2$ transforming this sequence into the sequence S':

$(aba_2ccb, aaa_3adb) \vdash_2 (aba_3adb, aaa_2ccb), (aba_1ccb, aba_3adb) \vdash_3$
$(aba_1adb, aba_3ccb), (bba_1cdc, aba_3ccb) \vdash_2 (bba_3ccb, aba_1cdc),$
$(bba_3ccb, aaa_1adb) \vdash_3 (bba_3adb, aaa_1ccb), (bba_3adb, bba_1cdc) \vdash_5$
$(bba_3adc, bba_1cdb), (aba_1adb, aaa_1ccb) \vdash_2 (aba_1ccb, aaa_1adb),$
$(aba_1ccb, aaa_1adb) \vdash_4 (aba_1cdb, aaa_1acb).$

Now S' is transformed into S'' for r previously defined. S'' is a B''-producing CTS in $CO_*(A'')$, where

$A'' = \{a_i x[r+2, |x|] | x \in A, 1 \leq i \leq m\}, \quad B'' = \{a_i z_i[r+2, |z_i|] | 1 \leq i \leq m\}$

For each $1 \leq i \leq m$ $card(ArbH(A'', a_i z_i[r+2, |z_i|]))$ is either $card(ArbH(A, z_i))$ or $card(ArbH(A, z_i)) - 1$.
Moreover, for each i such that

$$card(ArbH(A'', a_i z_i[r+2, |z_i|])) = card(ArbH(A, z_i)) - 1$$

there exist at least one step in S' where the strings exchange prefixes of length at most r. It follows that $lg(S'') \leq lg(S') - \lceil t/2 \rceil$, where

$t = card(\{i|card(ArbH(A'', a_i z_i[r+2, |z_i|])) = card(ArbH(A, z_i)) - 1\})$.
Consequently,

$$\begin{aligned} lg(S) &= lg(S') \geq lg(S'') + \lceil t/2 \rceil \geq \\ & \frac{\sum_1^m (card(ArbH(A'', a_i z_i[r+2, |z_i|])) - 1)}{2} + \\ & \lceil t/2 \rceil \geq \frac{\sum_1^m (Arbcard(H(A, z_i)) - 1)}{2}. \end{aligned}$$

□

Theorem 4 *There is a 2-approximation algorithm for computing the translocation distance from two sets of strings.*

Open problem: Is it possible to do it better?

Arbitrary crossover

(J. Kececioglu, R. Ravi 1995, S. Hannehalli 1995, S. Hannehalli, P. Pevzner 1995)

Let γ, δ two genomes having each n chromosomes with a total number m of genes.

Digraph $G_\gamma(\delta)$: m nodes and *continuous* and *discontinuous* arcs. Continuous arcs represent the right adjacency in the chromosomes of γ, discontinuous arcs represent left adjacency in the chromosomes of δ.

Formally, for any word $a_1 a_2 \dots a_p \in \gamma$ $G_\gamma(\delta)$ contains $p-1$ continuous arcs (a_i, a_{i+1}), $1 \leq i < p$. For any word $b_1 b_2 \dots b_q \in \delta$ $G_\gamma(\delta)$ contains $q-1$ discontinuous arcs (b_i, b_{i-1}), $1 < i \leq q$.

An *alternating cycle* in $G_\gamma(\delta)$; they are a uniquely determined partition of the set of arcs in $G_\gamma(\delta)$.

Let $\Psi(G_\gamma(\delta))$ the number of alternating cycles of $G_\gamma(\delta)$. Obviously, $\Psi(G_\delta(\delta)) = m - n$.

By applying a crossover to γ one ges γ_1 and $G_{\gamma_1}(\delta)$.

Lemma 5 *Let γ, δ two genomes with n chromosomes and m genes. Then*

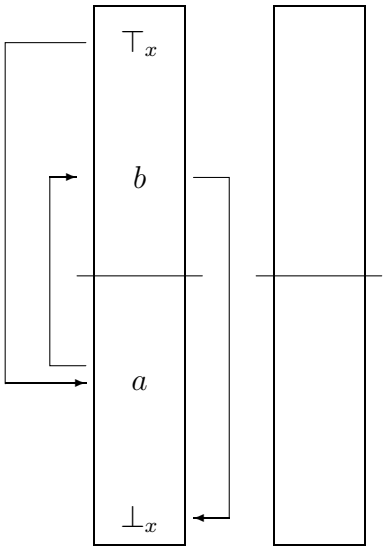
$$dc(\gamma, \delta) \geq m - n - \Psi(G_\gamma(\delta)).$$

Theorem 5 *There exists a 2-approximation algorithm for computing the distance between two genomes which runs in time $O(m^2)$.*

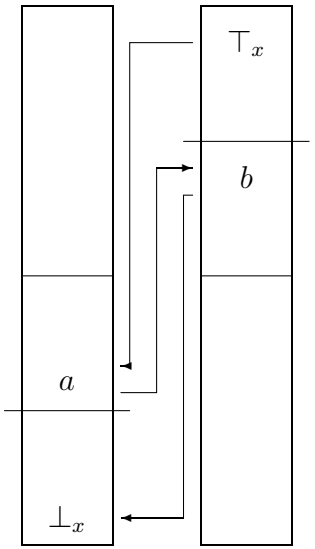
Proof.
program $Sort\_Crossover(\gamma, \delta)$;
begin
$\rho := \gamma$; $i := 0$;
while ρ *is not sorted* **do**
i:=i+1;
if *there exists a useful recombination* **then**
Choose a useful recombination σ_i *which permits further useful recombinations*
else *(Start a new phase)*
Find an inverted pair (a,b);
Choose a crossover σ_i *which put apart a and b in distinct words;*
endif;
$\rho := \rho \circ \sigma_i$;
endwhile;
end.

Fapt 1: *Each step of the algorithm contains at least three useful recombinations.*

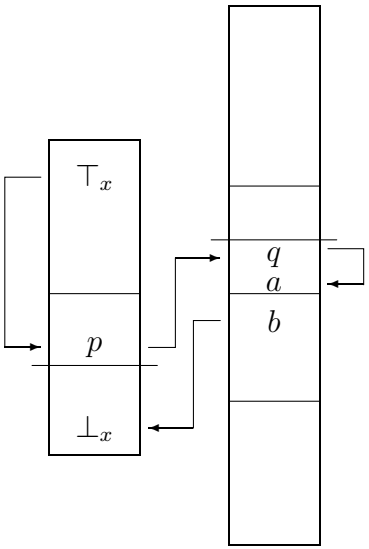
Let $\top_x$ and $\bot_x$ the starting and ending symbols of a word x din δ, respectively.



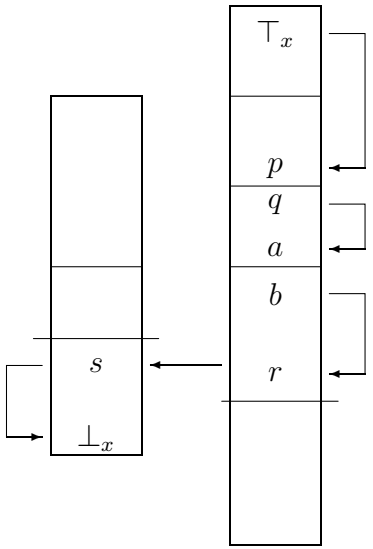
(a)



(b)



(c)



(d)

Fapt 2: *The algorithm transforms* γ *into* δ *in at most* $2(m - n - \Psi(G_\gamma(\delta))$ *recombinations.* $\square$

Recombination history
(J. Kececioglu, D. Gusfield 1994)

Pure recombination:

$$\begin{aligned} x &= v_1 v_2 \ldots v_{c+1}, \\ y &= u_1 u_2 \ldots u_{c+1}, \end{aligned}$$

recombined into $z = v_1 u_2 \ldots v_i u_{i+1} \ldots$.

Recombination with mutations:

$$z = \tilde{v}_1 \tilde{u}_2 \ldots \tilde{v}_i \tilde{u}_{i+1} \ldots$$

where $\tilde{v}_i$ and $\tilde{u}_i$ differ from v_i and u_i, respectively, by point mutations.

Cost: $d(v_1, \tilde{v}_1) + d(u_2, \tilde{u}_2) + \ldots + d(v_i, \tilde{v}_i) + d(u_{i+1}, \tilde{u}_{i+1}) + \ldots + c$.

The recombination distance for x, y, z is the minimal cost for producing z from x, y.

Given S the *recombination history of S* is a sequence of crossovers such that S is obtained from a *proto-pair*, each pair entering a crossover is avilable.

Problem: find the recombination history of a set S such that every recombination is less than d.

Let n be the total length of the words in S and k the number of words in S.

Unique crossover, pure recombination

Two phases: 1. Preprocessing: the words of S are analyzed for speeding-up the searching for a proto-pair.

Compute $PrefLen(x, y)$ and $SufLen(x, y)$ for every pair $x, y \in S$. This takes $O(n + k^2)$ time.

2. Each proto-pair is checked. The set of all words that can be obtained from x, y can be found by a "breadth-first searching. For every z in S that has not been obtained compute

$$MaxPrefLen(z) = max\{PrefLen(t, z)|t \text{ it was obtained}\},$$

$$MaxSufLen(z) = max\{SufLen(t, z)|t \text{ it was obtained}\}.$$

z can be obtained if

$$MaxPrefLen(z) + MaxSufLen(z) \geq |z|.$$

This takes $O(k)$ since there are $O(k)$ 'siruri. Adding it to the set of obtained words and up-dating $MaxPrefLen$ and $MaxSufLen$ takes $O(k)$. Therefore, one can check whether a candidate pair is a proto-pair in $O(k^2)$.

Theorem 6 *The pure recombination history by uniform unique crossover can be solved in time $O(n + k^3)$ and memory $O(n + k^2)$.*

Proof. 1. There are at most $k/2$ candidate pairs.

2. All candidates can be found in $O(n)$. □

Lemma 6 *Let $R(x, y)$ the subset of S obtained from x and y. If x, y is not a proto-pair, then no pair from $R(x, y)$ is proto-pair.*

Theorem 7 *The pure recombination history by unique crossover can be solved in time $O(n + k^4)$ and memory $O(n + k^2)$.*

Note that these algorithms find ALL proto-pairs. Is there any algorithm that finds ONE proto-pair faster that finding all.

Arbitrary recombination history by unique crossover

- $PrefCost(x, y, i)$ - edit distance of a prefix of y ending on the ith position from the prefix of x

- $SufCost(x, y, i)$ - edit distance of a suffix of y starting on the $i + 1$th position from the suffix of x

This takes $O(n^2)$ (dynamic programming).

During this computation we keep two values for each word not obtained yet

$$MinPrefCost(z,i) = min\{PrefCost(x,z,i)|x \text{ it was obtained }\},$$

$$MinSufCost(z,i) = min\{SufCost(x,z,i)|x \text{ it was obtained}\}.$$

z can be obtained if there exists i such that

$$MinPrefCost(z,i) + MinSufCost(z,i) \leq d.$$

Finding the next word takes $O(n)$. Adding and up-dating takes $O(n)$. If for any pair one must obtain $O(k)$ words, we have $O(kn)$. In the worst case, one must check $O(k^2)$ pairs.

Theorem 8 *Recombination history by unique crossover can be solved in $O(n^2 + k^3 n)$ time and $O(kn)$ memory.*

Crossover multiplu

Theorem 9 *Recombination history by multiple crossover can be solved in $O(n^3 + k^5)$ time and $O(l^2 + k^3)$ memory, where l is the length of the longest word in S.*

A generalization

Given S and two values d and c (maximal number of proto-words), does exist P of cardinality at most c such that P is a proto-set for S?

Theorem 10 *This is an NP-complete problem.*

It can be reduced to the "exact covering by 3-sets.