eSBH: An Accurate Constructive Heuristic Algorithm for DNA Sequencing By Hybridization

Yang Chen, and Jinglu Hu
Graduate School of Information, Production and Systems, Waseda University,
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka 808-0135, Japan,
Email: yangchen@ruri.waseda.jp, jinglu@waseda.jp

Abstract—Sequencing by hybridization is a promising cost-effective technology for high-throughput DNA sequencing via microarray chips. However, due to the effects of spectrum errors rooted from experimental conditions, a fast and accurate reconstruction of original sequences has become a challenging problem. In the last decade, a variety of analyses and designs have been tried to overcome this problem, where different strategies have different tradeoffs in speed and accuracy. Motivated by the idea that the errors could be identified by analyzing the interrelation of spectrum elements, this paper presents a new constructive heuristic algorithm, featuring an accurate reconstruction guided by a set of well-defined criteria and rules. The experiments on benchmark instance sets demonstrate that the proposed method can reconstruct long DNA sequences more accurately than current approaches in the literature.

I. INTRODUCTION

Since the last half of the twentieth century, in order to study the biological processes, and increase the understanding of organisms, researchers in molecular biology and bioinformatics have been engaging in analyzing and interpreting various types of biological data, one of which is the DNA sequence that contains genetic instructions used in the development and functioning of living things. As the knowledge of DNA sequences has become indispensable for biological research into why and how organisms live, it is naturally the first step to extract the data of DNA sequences from living things. A DNA sequence, representing the primary structure of a DNA molecule, is composed of a succession of letters \{A, C, G, T\} that denote four nucleotide bases {adenine, cytosine, guanine, thymine}.

The process of determining the order of nucleotide bases in a given DNA molecule is called DNA sequencing. Since the first DNA sequences were obtained in the early 1970s, the advent of DNA sequencing has significantly accelerated biological and biomedical research and discovery. Several sequencing strategies such as the chain-termination method [20] and shotgun methods [1] were developed for sequencing short DNA fragments and even for full genome sequencing, which have been successfully applied in the Human Genome Project (HGP) since the early 1990s to determine the sequence of human DNA and to understand the genetic makeup of human species.

Although these methods were widely used techniques in the past decade, a high demand for low-cost sequencing has driven the development of high-throughput technologies that parallelize the sequencing process [11]. Sequencing by hybridization (SBH) first envisioned nearly 20 years ago, is a non-enzymatic method that uses DNA microarray or gene chip to determine DNA sequences in the laboratory [12]. Recently, SBH has been rapidly evolving to be a useful and cost-effective approach for genomic resequencing [14], [16]. However, these advantages are currently offset by a few disadvantages, especially lower sequencing accuracy, given that all the research exploiting the information in DNA sequences relies to varying degrees on accurate raw data.

II. RELATED WORKS

In fact, a low sequencing accuracy of SBH can be traced back to its two-stage working process. The SBH method first detects via a microarray chip, all short ‘words’ of nucleotides (or oligonucleotides) of a given length \(l\) that make up a DNA sequence (of known length \(n\), \(n > l\)). The hybridization experiment is a biochemical process conducted under carefully controlled conditions (e.g., temperature, pH, salt concentration) [12]. These \((n - l + 1)\) individually detected DNA words then define a set called spectrum, which are combined to reconstruct a long “sentence”, namely the original sequence.

It was reported in [18] that the reconstruction problem can be reduced to an Eulerian path problem solved in polynomial time with an exact algorithm. An example was found in [5].

Unfortunately, SBH is not so ideal that the hybridization experiment may bring errors into the spectrum, because DNA-word recognition can be error prone when hybridization conditions are improper for specific oligonucleotides. Usually, there exist two types of spectrum errors: (1) negative errors (i.e., missing oligonucleotides that should appear), and (2) positive errors (i.e., false oligonucleotides that should not appear). These errors in the spectrum cause the reconstruction problem being a strongly NP-hard combinatorial one [5], which means accurate DNA sequences cannot be obtained in practice.

Current approaches that deal with the SBH experimental errors may be mainly divided into two categories. Some seek to reduce the number of errors in the biochemical part by presenting isothermic oligonucleotide libraries [6] and an analog-spectrum model [19], as well as their corresponding reconstruction algorithms. On the other hand, many researchers focus on the combinatorial part of SBH, trying to reconstruct the original sequence from the spectrum containing errors, where some earlier research considers simply a restricted model of errors [18], [15], [2], for example, only one type of errors. The more general case where both types of errors are allowed has been studied in the last decade, since Blazewicz et al.'s work where it was reduced to a variant of the selective traveling salesman problem (TSP) that was then solved by a branch and bound algorithm [3].

Having practiced the traditional lines of attack for NP-hard problems where exact algorithms work reasonably fast only for relatively small problem scales, the research community has devoted themselves to devising heuristic algorithms for SBH reconstruction. An overview on existing methods for the SBH problem was listed in [9]. They could be further partitioned into two classes: (1) constructive heuristics [4], [22], [8];
and (2) metaheuristics such as tabu-and-scatter-search [7], genetic algorithms [13], [10], and most recently, ant colony optimization [9]. Both methods have their own advantages and disadvantages. The constructive heuristics reconstruct sequences in quite short time. However, since they are developed based on greedy strategies, the sequencing accuracy decreases when handling long sequences. The metaheuristics, on the contrary, despite that they achieve accurate reconstruction even for long sequences, they have to sacrifice more time in iterative search inevitably.

In this paper, we present a new constructive heuristic algorithm called eSBH in order to improve the sequencing accuracy for long DNA sequences when reconstructed from the spectra with errors, without decreasing the sequencing speed too much. The main idea of this algorithm is, to identify spectrum elements and determine their possible locations in a reconstructed sequence, by a set of criteria and rules based on the relationship between spectrum elements. This helps to alleviate the effects from spectrum errors and repetitions in original sequences. The rest of this paper is organized as follows. Section III introduces our algorithm from studying the effects of spectrum errors. In Section IV, the experimental results are analyzed and explained to evaluate its effectiveness. We summarize the work and suggest some further research directions in Section V.

III. THE eSBH ALGORITHM

A. The Effects of Spectrum Errors

Before presenting our constructive heuristic, let us first take an insight into the effects of positive and negative errors within the spectrum on the reconstruction of the original sequence. We assume that a spectrum $S = \{1, ..., m\}$, comprising $m$ (called spectrum size) oligonucleotides of length $l$ (as the input to an algorithm) was obtained from a DNA microarray chip via hybridization experiment. The goal is to reconstruct from $S$, a sequence $s_t$ that is expected to be the original one $s_t$, of length $n$ (as the output from an algorithm).

Definition 1. Let $i, j \in S$ ($i \neq j$) be two oligonucleotides of length $l$. The **maximal overlap length** from $i$ to $j$, $\alpha_{ij}$ is $7$ ($l = 10$)

![Fig. 1. The maximal overlap length from $i$ to $j$, $\alpha_{ij} = 7$ ($l = 10$)](image)

The presence of spectrum errors makes the reconstruction problem harder, but it may still be solved by managing these errors provided that $e^+$ and $e^-$ are relatively low (e.g. $< 20\%$). In fact, if we could identify positive errors in $S$ and exclude them, or tolerate negative errors by allowing $\alpha_{ij} < l - 1$ during reconstruction, then the effects of both errors could to some extent be relieved.

With these in mind, we are ready to introduce the algorithm. On the whole, the eSBH is a two-stage algorithm. The first stage is to construct from $S$, accurate subsequences of the original sequence, by a constructive heuristic as described in Sections III-B and III-C. If more than one subsequences are constructed in the first stage, then they are combined into one sequence in the second stage, by a branch and bound algorithm as described in Section III-D. The pseudo-code of eSBH is shown in Algorithm 1, which is described in detail as follows (for the real code, please visit: http://sites.google.com/site/yangchen2005/).

### Algorithm 1 The eSBH Algorithm

**Require:** A spectrum $S = \{1, ..., m\}$.

**Ensure:** Reconstructed sequences $s_r$.

1: **STAGE 1:**
2: $H = \{h \mid \|\text{pre}(h)\| = 0 \land |\text{suc}(h)| \neq 0, h \in S\}$, $H' = \emptyset$
3: for all $h \in H$ do
4: $(s_r(h), H') \leftarrow \text{BuildSubsequence}(h)$
5: end for
6: while $H' \neq \emptyset$ do
7: for all $h' \in H'$ do
8: $(s_r(h'), H'') \leftarrow \text{BuildSubsequence}(h')$
9: end for
10: $H'' \leftarrow \text{FindIsolatedSubcycle}(S)$
11: end while

12: **STAGE 2:**
13: if $|H \cup H'| = 1$ then
14: return $\{s_r(i) \mid i \in H \cup H'\}$
else
15: RemoveRedundantSubseq($\{s_r(i) \mid i \in H \cup H'\}$)
16: $P \leftarrow \text{EnumerateFeasiblePermutation}(\{s_r(i)\})$
17: for all $p \in P$ do
18: $(s_r(p), m(p)) \leftarrow \text{CombineIntoOneSequence}(p)$
19: end for
20: return $\{s_r(p^*) \mid p^* = \arg \max \{m(p) \mid p \in P\}\}$
21: end if

B. Starting Construction from Heads

A intuitive idea is to begin reconstructing sequence $s_r$ with a correct oligonucleotide as its head, say $i \in S$, which is expected to be the head of $s_t$. The head $i$ is selected from the spectrum $S$ that contains oligonucleotides of various identities, which could be parts of $s_t$ (head, body and tail), or even positive errors. In fact, the identity of an oligonucleotide $i \in S$, may be distinguished from examining $i$’s neighbors in $S$, i.e., its predecessors and successors.

Definition 2. Let $i, j \in S$ ($i \neq j$) be two oligonucleotides of length $l$. The **predecessors and successors** of $i$ are

$$\text{pre}(i) = \{j \mid (l - 1) - \epsilon \leq \alpha_{ji} \leq l - 1\},$$

$$\text{suc}(i) = \{j \mid (l - 1) - \epsilon \leq \alpha_{ij} \leq l - 1\}$$
respectively, where $\epsilon$ is a fault-tolerant factor. In particular, $j^\ast$, $j^+\ast$ are called the best predecessor and best successor of $i$ if and only if $j^\ast \in \arg\max\{o_j, \ |j \in \pre\{i\}\}$, $j^+\ast \in \arg\max\{o_j, \ j \in \suc\{i\}\}$ respectively.

The fault-tolerant parameter $\epsilon (0 \leq \epsilon \leq l - 1)$ is a relaxation parameter used to relieve the effects of both types of errors. A large $\epsilon$ helps to tolerate continuous appearances of negative errors, while a small $\epsilon$ helps to exclude positive errors as well as repetitive oligonucleotides in $S$. Hence, the value of $\epsilon$ is related to $\epsilon^- = \epsilon^+$ and the repetitions in $S$, reaching a trade-off to manage both types of errors for a specific sequence (settings given in Section IV). From the predecessors and successors of an oligonucleotide $i \in S$, one can infer in a rough degree, whether $i$ is a positive error, or $i$ belongs to the head, body, or tail of $s_t$. Table I listed the rough inferences of $i$’s identity. In this way, many positive errors in $S$ could be filtered. Moreover, those oligonucleotides with no predecessors but at least one successors are selected out of $S$, say

$$H = \{h \mid \pre(h) \neq 0 \text{ and } \suc(h) \neq 0, \ h \in S\},$$

as candidate heads from which $s_t$ is to be reconstructed. It is worth mentioning here: (1) despite that some $h \in H$ may actually be positive errors, they could eventually be discriminated; (2) usually $H$ includes more than one heads, due to the negative errors in $s_t$’s body, which prompts us first to build several short subsequences rather than the direct construction of a long sequence.

C. Subsequences Building

The initial stage produces a set of candidate heads $H \subseteq S(\#H > 0)$ as the bases for further construction. In this stage, from each head $h \in H$, a short sequence $s_t(h)$ is going to be built by extending it with other oligonucleotides in $S$ as its body, which is expected to be a subsequence of $s_t$. All $|H|$ subsequences are generated one by one, where each element in $S$ could be linked in at most once.

BuildSubsequence($h$): The construction of $s_t(h)$ is an iterative process, in each step of which an oligonucleotide $i$ chosen from $S$ is linked to the end of $s_t(h)$ in their maximal overlap length, until $s_t(h)$ can no longer be extended. If all these linkings are correct, the construction could be error-free. Let us consider a typical $(k + 1)$-th step, when $s_t(h)$ has been partially built by linking $(i_1, \ldots, i_k)$. We select an $i_{k+1}$ from $\suc(i_k)$ by referring to three criteria: availability, before-consistency, and after-extensibility. Let $j \in \suc(i_k)$ be an oligonucleotide from $S$.

- **Criterion I:** $j$ is available if and only if (a) $j$ has not yet been linked in constructing $\{s_t(h) \mid h \in H\}$, and (b) $s_t(h)$ will not exceed the expected length $n$ if $j$ is linked to its end (i.e. $i_k$). An unavailable $j$ is never selected.

- **Criterion II:** $j$ is before-consistent if and only if (a) $j$ has $i_k$ as its only best predecessor, or (b) $j$ has at least two best predecessors whose nucleotide bases are all the same as $i_k$. The $j$ which is not before-consistent, even if available, cannot be selected since the inconsistency of $j$ may bring an incorrect linking ($i_k, j$) in $s_t(h)$. This helps to recognize repetitive oligonucleotides from $S$.

- **Criterion III:** $j$ is after-extensible if and only if $|\suc(j)| \neq 0$. This helps to discriminate positive errors from $S$, but note that $j$ which is not after-extensible may also be selected in case of the $s_t(h)$’s tail.

Based on these criteria, we design a set of rules as guidelines for linking a correct $i_{k+1}$ while excluding improper ones. All these rules compose a procedure (Rule A,...,Rule F) that is performed iteratively till $s_t(h)$ is constructed.

- **Rule A:** If there is no available $j \in \suc(i_k)$, then stop the extension of $s_t(h)$, and start constructing from another candidate head in $H$.

- **Rule B:** If there are available $j \in \suc(i_k)$, but none of them are before-consistent, then stop extending $s_t(h)$ and save those after-extensible $j$ (if exist) as hidden heads in a set $H’$.

The hidden head in $H’$, different from a candidate head in $H$, could have predecessors (e.g. $i_k$). Instead of being selected before reconstruction, a hidden head may be generated on demand when extending subsequences (e.g. $s_t(h)$), usually in two cases. First, there exist in $\suc(i_k)$ at least two equal alternatives matching the current extension so that one cannot decide, where the correct $i_{k+1}$ is confused by some repetitive oligonucleotides in $S$ (see Rules B and F). Secondly, some $j \in \suc(i_k)$ mismatching the current extension may be an alternative for the coming ones (see Rules B and C), which is also reserved in $H’$ to prevent so-called isolated subcycles treated later.

- **Rule C:** If there is only one $j \in \suc(i_k)$, both available and before-consistent, then it is accepted as $i_{k+1}$ and linked to $i_k$. Besides, the other available $j$ with after-extensibilities (if exist) are saved as hidden heads.

- **Rule D:** If there are at least two $j \in \suc(i_k)$, both available and before-consistent, but none of them are after-extensible, then the $j$ with the maximal $o_{i_k, j}$ is picked as $i_{k+1}$. This means that $i_{k+1}$ becomes the tail of $s_t(h)$.

It is mentioned that in Rule D, a positive error may be accepted as $i_{k+1}$ by mistake, but its effect is smaller than that of incorrect linkings in bodies. Section III-D involves managing positive errors at the tail of a subsequence.

- **Rule E:** If there is only one $j \in \suc(i_k)$ that is available, before-consistent and after-extensible, then it is accepted as $i_{k+1}$ and linked to $i_k$. It usually happens when $i_{k+1}$ becomes a body of $s_t(h)$.

- **Rule F:** If there are at least two $j \in \suc(i_k)$ that are available, before-consistent and after-extensible, then all of them are saved as hidden heads and the current extension stops, unless their nucleotide bases are all the same as each other, only in which case do we pick any $j$ as $i_{k+1}$ and continue extending $s_t(h)$.

Accordingly, all short sequences $\{s_t(h) \mid h \in H\}$ are built by adopting the above procedure. However, this may leave a set of hidden heads $H’$ to be handled. In fact, we treat hidden heads in the same way as candidate heads. For each hidden head $h’ \in H’$, if it is available, then a short sequence $s_t(h’)$ is extended from $h’$. In extending $s_t(h’)$, new hidden heads may emerge and then be queued in $H’$ iteratively, till all such heads in $H’$ are disposed.

FindIsolatedSubcycle($S$): These short sequences, as previously stated, are expected to cover the entire $s_t$, from which
could be assembled. However, it is found that a small part of\( s_t \) may fail to be covered. In fact, this part is linked up from oligonucleotides that constitute an unexpected cycle (in terms of neighbors) out of the reach from \( h \in H \) and \( h' \in H' \). Hence, we call it an isolated subcycle, which is attributed to both negative errors and repetitions in \( s_t \). In order to discover

isolated subcycles, every available oligonucleotide, say \( j \in S \), is checked if it could be linked from \( i_k \), the tail of constructed subsequences \( s_r(h) \) or \( s_r(h') \), so that

\[
(l - 1) - \epsilon' \leq o_{ik,j} < (l - 1) - \epsilon,
\]

where \( \epsilon' = \epsilon + \Delta(0 < \Delta \leq l - 1) \). This means that \( \Delta \epsilon \) (settings in Section IV) more continuous negative errors are allowed to break the isolation, so that the subcycle where \( j \) is located can be reached from \( i_k \), as illustrated in Fig. 2. The \( j \) is then added in \( H' \) and treated as a hidden head in the same way as before.

**D. Combining Into A Whole Sequence**

It is now ready to combine all the subsequences \( \{s_r(i) \mid i \in H \cup H'\} \) produced in the previous stage into a whole sequence \( s_r \), which is expected to be same as the original one \( s_t \). In fact, this stage could suffer more from the effect of spectrum errors, which causes the combining problem more complex than subsequence building. However, it is observed that the problem size has already been greatly reduced, from the spectrum size \( m \) to the number of subsequences \( |H \cup H'| \). With the aim at high sequencing accuracy, a branch and bound algorithm is therefore carefully devised to search for the optimal combination of \( s_r \).

**RemoveRedundantSubseq(\( s_r(i) \))**: Considering the time complexity of branch and bound largely depends on the problem size, we first attempt to reduce the number of subsequences by removing redundant ones, which are verified in line with two measures. One is the number of oligonucleotides \( m(i) \) linked in a subsequence \( s_r(i) \). The other one is with regard to the best predecessor and successor of \( s_r(i) \) defined as follows.

**Definition 3.** Let \( s_r(i) \) and \( s_r(j) \) (\( s_r(i) \neq s_r(j) \)) be two subsequences built up by \( \{i_1 = i, \ldots, i_{m(i)}\} \) and \( \{j_1 = j, \ldots, j_{m(j)}\} \) respectively. The maximal overlap length from \( s_r(i) \) to \( s_r(j) \) is

\[
O_{s_r(i),s_r(j)} = \max \left\{ o_{m(i) - k(i), j_1 + k(j)} \mid 0 \leq k(i), k(j) \leq \delta \right\},
\]

where \( \delta (0 < \delta < \min(m(i), m(j))) \) is another fault-tolerant factor that helps to manage positive errors, usually at the two ends of a subsequence. The larger \( \delta \) is, the more continuous appearances of positive errors at the tail of \( s_r(i) \) and at the head of \( s_r(j) \), could be excluded when combining these two subsequences. The \( \delta \) is assigned according to \( e^r \), whose settings are given in Section IV.

**Definition 4.** Let \( s_r(i) \) and \( s_r(j) \) (\( s_r(i) \neq s_r(j) \)) be two subsequences. The predecessors and successors of \( s_r(i) \), are those \( s_r(j) \) with the maximal or second maximal \( O_{s_r(i),s_r(j)} \) respectively. In particular, \( s_r(j^*) \), \( s_r(j'^*) \) are called the best predecessor and best successor of \( s_r(i) \) if and only if \( s_r(j^*) \in \arg \max\{O_{s_r(i),s_r(j)}\} \), \( s_r(j'^*) \in \arg \max\{O_{s_r(i),s_r(j)}\} \) respectively.

A subsequence \( s_r(i) \) is then said to be redundant if (1) \( m(i) \leq \delta \), and (2) either of the following

\[
O_{s_r(i),s_r(i)} < (l - 1) - \epsilon' \text{ or } m(j^*) \leq \delta,
\]

holds, where \( s_r(j^*) \) and \( s_r(j'^*) \) are the best predecessor and best successor of \( s_r(i) \) respectively. Intuitively, a redundant subsequence is a short subsequence that cannot closely link with long subsequences from either ends, which is removed as it may be built up accidentally by at most \( \delta \) positive errors.

**EnumerateFeasiblePermutation(\( s_r(i) \))**: Next we apply a branch and bound algorithm to enumerate feasible permutations \( P \) of subsequences, in which way they are to be combined into a whole sequence. This is similar to building a subsequence by linking oligonucleotides, but differs in selecting head and body components.

- The first branching occurs when an empty permutation \( p \) starts from a head subsequence \( s_r(i) \) with the minimal or second minimal \( O_{s_r(i),s_r(i)} \), where \( s_r(j^*) \) is a best predecessor of \( s_r(i) \).
- Then, a general \( (k + 1) \)-th branching step is considered when \( p = (s_r(i_1), \ldots, s_r(i_k)) \). The \( s_r(i_{k+1}) \) is selected from the successors of \( s_r(i_k) \) that are both available and before-consistent. The two criteria here are simplified as (I') \( s_r(j) \) is available if \( s_r(j) \notin \{s_r(i_1), \ldots, s_r(i_k)\} \), and (II') \( s_r(j) \) is before-consistent if \( s_r(j) \) has a predecessor \( s_r(i_k) \).

The branching rule is applied recursively till all permutations are generated. Since most non-redundant subsequences should be present in \( p \), one performs the bounding rule: if the number of subsequences absent from \( p \) is less than a threshold \( \delta_c \), then \( p \) is said to be feasible. Hence, the feasible \( p \) are saved into \( P \), while the infeasible ones are discarded.

**CombineIntoOneSequence(\( p \))**: Finally, a sequence \( s_r \) is constructed based on a feasible \( p \in P \) respectively. Created with \( s_r(i_1) \), \( s_r \) is built up by subsequences in the order indicated by \( p \), with adjacent pairs \( \{s_r(i_k), s_r(i_{k+1})\} \) linked in their maximal overlap length. The \( s_r \) is extended as long as it does not exceed the expected length \( n \). Once it is over-length, one tries to remove at most \( \delta \) oligonucleotides at either the head or the tail of \( s_r \), since they may be positive errors as described earlier. After all sequences are built, the best ones are selected according to an objective function. Let \( s_r(p) \) be a sequence built based on a feasible \( p \in P \), and \( m(p) \) be the total number of oligonucleotides to build \( s_r(p) \). The resultant reconstructed sequences are

\[
s_r \in \{s_r(p^*) \mid p^* = \arg \max \{m(p) \mid p \in P\}\},
\]

which are expected to resemble the original sequence \( s_t \) (note that one may have several reconstructed sequences rather than only one as the outcome of eSBBH).
IV. RESULTS AND DISCUSSION

The new algorithm eSBH, is implemented by MATLAB 7.9.0, and the computational experiments are carried out on a PC with a Core Duo 1.8GHz CPU, 1.0GB RAM, and Ubuntu 9.10 OS. In order to evaluate the effectiveness of eSBH, we employ two popular benchmark instance sets from [7]: (i) real DNA sequences coding human proteins taken from GenBank (denoted by IS1), and (ii) random DNA sequences generated according to the uniform distribution (IS2). Both IS1 and IS2 contain original sequences of various lengths \( n \in \{109, \ldots, 509\} \) (step 100), where for each length 40 instances are presented. The spectrum is then generated via a simulated hybridization experiment with probe length \( l = 10 \), where \( \epsilon^{-} = 20\% \) negative errors (20 missing oligonucleotides in case of \( n = 109 \)) and \( \epsilon^{+} = 20\% \) positive errors randomly generated are introduced.

There are three integer parameters with eSBH: \{\( \epsilon, \delta, \Delta \epsilon \). Note that the number and range of parameters are smaller than most metaheuristics in the literature. In this experiment, they are set as \( (\epsilon, \delta, \Delta \epsilon) = (2, 2, 2) \) for both IS1 and IS2 based on the preliminary experience. In fact, these settings could influence the performance of reconstruction in terms of accuracy and speed. As an example, for various \( (\epsilon, \delta, \Delta \epsilon) \) in Table II, we test the eSBH on 40 instances of length \( n = 509 \) in IS1 and the average results over all instances are summarized.

- The solution quality, as introduced in [7], is the number of spectrum elements composing reconstructed sequences \( s_{\text{r}} \), with a maximum of \((n-l+1) \times (1-\epsilon^{-})\). The solved instances is the number of instances with the maximal solution quality.
- The global and local similarity mean the resemblance between the reconstructed sequence \( s_{\text{r}} \) and the original one \( s_{\text{t}} \), which could be good measures of sequencing accuracy (in case of more than one \( s_{\text{r}} \), only the maximal similarity is listed). These two similarities are calculated via global/local pairwise alignment algorithms [17], [21] respectively with a scoring scheme \((\text{match, mismatch, gap}) = (1, -1, -1)\).
- In the bottom row, the computation time indicates the speed of sequence reconstruction.

It is learned from this case that, (i) biasing \( \epsilon \) or \( \delta \) away from 2, especially when decreasing them to 1, may result in a less accurate and slower reconstruction; while (ii) there is a relatively small effect of varying \( \Delta \epsilon \) on the results.

Tables III–IV show the performance of eSBH on IS1 and IS2 respectively, where the spectrum size \( m \) indicates the problem scale, which is calculated as \( m = (n-l+1) \times (1-\epsilon^{-} + \epsilon^{+}) \). For each spectrum size, the eSBH is applied to 40 instances and the summarized results are listed. As can be learned from Table III, the proposed eSBH performs well on real DNA sequences in IS1, achieving higher accuracy (nearly 100\%) for not only short sequences but the long ones as well. However, incomplete reconstructions are found on a few instances, due to an isolated subcycle formed at the head of sequences that cannot be tackled by eSBH. Besides, Table IV shows a slight decrease in accuracy on IS2. Further investigations reveal that this results from an insufficient “Definition 4” where parameters may be introduced. On the other hand, eSBH performs the reconstruction very fast, working out an accurate sequence from a size-500 spectrum in 7 seconds on a PC.

For a further validation on the performance of eSBH for real DNA sequences, we compare it with some selected state-of-the-art algorithms for SBH reconstruction in the literature, including two constructive heuristics: the one (identified by OW) in [4], HSM in [8]; and two metaheuristics: tabu-and-scatter-search (TSS) in [7], multi-level ant colony optimization (ML-ACO) in [9]. These algorithms are compared by two measures: (i) the local similarity that terms the reconstruction accuracy, and (ii) the computation time that terms the reconstruction speed. It should be mentioned that due to the availabilities of other algorithms except TSS, all the data are directly quoted from their original papers, which could mean that the comparison on computation time may be unfair since they are obtained under different experimental conditions. As shown in Fig. 3, for small spectrum sizes, all these algorithms achieve accurate reconstructions. However, the accuracy curves diverge as the spectrum size increases, where most algorithms including HSM degrade their performance to some extent, while the eSBH strives to maintain a high sequencing accuracy. On the other hand, the proposed algorithm works much faster than TSS, and its time overhead grows moderately with the increasing spectrum sizes compared with that of ML-ACO. These results demonstrate that the eSBH could be an effective approach to the accurate reconstruction of long DNA sequences.

![Fig. 3. Comparisons on the performance of various algorithms for IS1. (*)](#)
V. CONCLUSIONS

This work presents a new constructive heuristic algorithm for the reconstruction problem of DNA sequencing by hybridization with spectrum errors. The eSBH algorithm can achieve higher accuracy in reconstruction from a large spectrum, than other constructive heuristics and some metaheuristics in the literature, especially for real DNA sequences in the benchmark instance sets.

Further efforts will be made to investigate the performance on other sequencing conditions (e.g. $n > 500$, $l < 10$, $c^2/e^c > 20\%$). Besides, the problem discussed in Section IV suggest a two-way construction started from the bodies for higher accuracy. Some metaheuristics may also be hybridized to speed up the combination. The authors will henceforth compare with the methods for a non-standard library or spectrum [6], [19]. We hope all these efforts could help SBH become a practical technology for next-generation DNA sequencing in the new century.

ACKNOWLEDGMENT

The authors would like to thank J. Blazewicz and M. Kasprzak for providing their tabu-and-scatter-search program and benchmark instance sets.

This research was supported by “Japanese Government (MEXT) Scholarship Program” and “Ambient SoC Global COE Program of Waseda University” of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

REFERENCES


