DNA Sequence Compression Using Adaptive Particle Swarm Optimization-Based Memetic Algorithm

Zexuan Zhu, Jiarui Zhou, Zhen Ji, *Member, IEEE,* and Yu-Hui Shi, *Senior Member, IEEE*

***Abstract*—With the rapid development of high-throughput DNA sequencing technologies, the amount of DNA sequence data is accumulating exponentially. The huge influx of data creates new challenges for storage and transmission. This paper proposes a novel adaptive particle swarm optimization-based memetic algorithm (POMA) for DNA sequence compression. POMA is a synergy of comprehensive learning particle swarm optimization (CLPSO) and an adaptive intelligent single particle optimizer (AdpISPO)-based local search. It takes advantage of both CLPSO and AdpISPO to optimize the design of approximate repeat vector (ARV) codebook for DNA sequence compression. ARV is first introduced in this paper to represent the repeated fragments across multiple sequences in direct, mirror, pairing, and inverted patterns. In POMA, candidate ARV codebooks are encoded as particles and the optimal solution, which covers the most approximate repeated fragments with the fewest base variations, is identified through the exploration and exploitation of POMA. In each iteration of POMA, the leader particles in the swarm are selected based on weighted fitness values and each leader particle is fine-tuned with an AdpISPO-based local search, so that the convergence of the search in local region is accelerated. A detailed comparison study between POMA and the counterpart algorithms is performed on 29 (23 basic and 6 composite) benchmark functions and 11 real DNA sequences. POMA is observed to obtain better or competitive performance with a limited number of function evaluations. POMA also attains lower bits-per-base than other state-of-the-art DNA-specific algorithms on DNA sequence data. The experimental results suggest that the cooperation of CLPSO and AdpISPO in the framework of memetic algorithm is capable of searching the ARV codebook space efficiently.**

***Index Terms*—Approximate repeat vector, DNA sequence compression, memetic algorithm, particle swarm optimization.**

Manuscript received April 30, 2010; revised September 29, 2010 and March 13, 2011; accepted April 26, 2011. Date of current version September 30, 2011. This work was supported in part by the National Natural Science Foundation of China, under Grants 60872125, 61001185, and 60975080, in part by the Fok Ying-Tung Education Foundation, Guangdong Natural Science Foundation, under Grant 10151806001000002, in part by the Foundation for Distinguished Young Talents in Higher Education of Guangdong, under Grant LYM10113, and in part by the Shenzhen City Foundation for Distinguished Young Scientists.

Z. Zhu and Z. Ji are with the Shenzhen City Key Laboratory of Embedded System Design, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: jizhen@szu.edu.cn).

J. Zhou is with the College of Biomedical Engineering and Instrument Science, Zhejiang University, Hangzhou 310027, China.

Y.-H. Shi is with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2011.2160399

I. Introduction

DNA IS A biological macromolecule with double helix structure storing hereditary information of a life. Understanding the information encoded in DNA is essential for basic biological and medical research. The information of DNA is mainly represented as the order of the nucleotides that can be detected with DNA sequencing methods. The rapid advancement of high-throughput sequencing technologies, such as next generation sequencing, have dramatically reduced the cost of sequencing [1]–[3] and a large number of DNA sequencing projects have been carried out to obtain the DNA sequences of various organisms. The exponentially increasing amount of DNA sequencing data has posed big challenges on storage and transmission. DNA sequence compression has become a crucial task for managing, modeling, and learning about sequences.

DNA sequences consist of a series of symbols representing the nucleotide bases: Adenine (A), Thymine (T), Guanine (G), and Cytosine (C). They can be stored in the form of text with two binary bits encoding each of the four bases, i.e., two bits-per-base (BPB), whereas the sequences are naturally abundant in repeated fragments, which make it possible to compress them in less than two BPB. General-purpose compression methods cannot achieve satisfying performance on DNA sequences, as they do not take account of the intrinsic biological characteristics of DNA. New techniques tailored for DNA sequences are thus required to accomplish substantial compression.

Many DNA-specific compression methods have been invented and shown to have promising compression performance [4]. Most of the existing DNA-specific methods compress sequences by substituting repeated subsequences with a convenient coding scheme or with short pointers to an entry in a reference dictionary/codebook. As illustrated in Fig. 1, with a well-designed codebook, the repeated fragments "ATCCG" in a sequence can be represented by the corresponding code vector index *i* taking only few bits. The storage space can be considerably reduced, even thought an extra space is required for the codebook. Substitutional methods are simple yet highly effective for DNA sequence compressing.

This paper focuses on substitutional methods in vertical mode, which compresses multiple sequences as a whole by referring to information contained in the entire set. We first propose a novel approximate repeat vector (ARV) model to

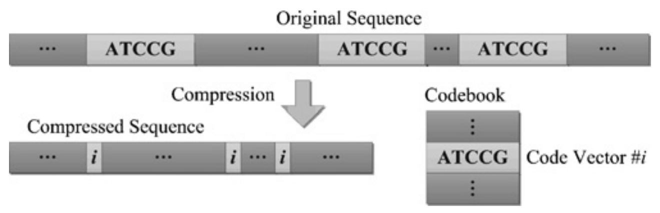


Fig. 1. Compression with codebook.

represent the four repeat patterns (i.e., direct, mirror, pairing, and inverted repeats) in DNA sequences with approximate matching mechanism. Based on the ARV model, the most commonly repeated fragments across multiple sequences are identified to form a reference codebook and duplicate instances in the sequences are then substituted with reference to the corresponding ARV. The design of ARV codebook is an optimization problem to maximize the exact cover rate and minimize the base distance of the code vectors on the sequences.

To solve the aforementioned optimization problem, we further introduce a novel adaptive particle swarm optimization-based memetic algorithm (POMA). To the best of our knowledge, this is the first study of evolutionary computation techniques on DNA sequence compress problem. The new algorithm hybridizes the comprehensive learning particle swarm optimizer (CLPSO) [5], [6] with an adaptive intelligent single particle optimizer (AdpISPO). Particle swarm optimization (PSO) [7] is a metaheuristics technique that has been shown to search well on complex optimization problems [8]–[15]. It has attracted much interest mainly due to its high efficiency search without any need for gradient information. However, PSO suffers from the problem of premature convergence and slow-down of the swarm when approaching local optima [14]. CLPSO addresses the issue of premature convergence with the introduction of the comprehensive learning strategy. To mitigate the slow-down in the swarm search, an AdpISPO-based local search to fine-tune the swarm particles is introduced here. AdpISPO optimizes a particle in each dimension and complements CLPSO with exploitative capabilities in neighborhood of particles. The synergized CLPSO and AdpISPO thus forms a memetic algorithm (MA) [16] and labeled here as POMA. The search performance of the proposed POMA is compared with other state-of-the-art counterpart algorithms on 29 benchmark functions and 11 DNA sequence datasets. Experimental results showed that POMA is able to obtain better or competitive performance on both the benchmark functions and realistic DNA sequence data.

The rest of this paper is organized as follows. Section II provides some background knowledge and literature reviews on DNA sequence compression and MA. Section III presents the details of the proposed ARV model. Section IV describes the proposed POMA including the CLPSO-based global search and the AdpISPO-based local search. Section V presents the experimental study conducted on the benchmark functions and DNA sequence data, followed by Section VI discussing the running time issue and future work. A brief conclusion is then given in Section VII.

II. Background and Literature Review

In this section, we present some background knowledge and literature reviews on the DNA sequence compression problem and MA.

A. DNA Sequence Compression

The purpose of DNA sequence compression is to find an efficient encoding method and reduce the space to store the exponentially increasing sequence data. DNA compression must be lossless to retain all genetic information enciphered in the sequences and to guarantee the reliability of the raw data. DNA sequences are commonly recorded in the form of text, however traditional text compression techniques (e.g., bzip2, gzip, and *compress*) failed in compressing them efficiently [17]. Unlike common text data, DNA sequences contain an abundance of repeated fragments, which could occur at long intervals and in peculiar patterns. The intrinsic characteristics of DNA sequences have thus led to the introduction of specialized compression algorithms.

To date, many specialized DNA sequence compression algorithms have been proposed in the literature. One of the most commonly used DNA sequence compression technique is substitutional compression, which compresses sequences by substituting repeated subsequences with a convenient or specially designed coding scheme. Grumbach and Tahi [18] proposed the first paradigm of substitutional method, known as BioCompress, to compress the exact DNA subsequence repeats using Fibonacci coding. BioCompress represents one of the earliest attempts to consider the specificity of genetic information, such as the present of palindromes for DNA sequence compression. Grumbach and Tahi improved the algorithm to BioCompress-2 in [19] by introducing Markov model to encode the non-repeated regions. Based on BioCompress-2, Chen *et al.* exploited the approximate repeats to arrive at GenCompress [20] and DNACompress [21] as extensions. The studies in [20] and [21] suggested that approximate repeat also play an important role in DNA sequence compression. Matsumoto and Tahi [17] proposed a CTW+LZ algorithm that searches for approximate repeats and palindrome using hash and dynamic programming. CTW+LZ was shown to achieve promising compression rate, but at the cost of significantly increased computational complexity. Korodi and Tabus [22] presented the GeNML to combine substitution and specific normalized maximum likelihood model in fragments compression. GeNML was demonstrated to achieve much higher compression ratio than many other DNA compression algorithms within a fixed execution time. Substitutional methods are generally simple yet they are not devoid of compression efficiency on large datasets [23].

Another popular DNA sequence compression technique is statistical compression, which typically constructs a probabilistic model for predicting the generation of symbols and subsequently encodes the symbols using arithmetic coding routines based on the prediction. For example, CDNA [24] and ARM [25] obtain the distribution of each symbol from approximate partial matching and summation of probabilities over all explanations, respectively. XM [26], which is well established as the best single sequence compression algorithm recently [4], [27], estimates the probability distribution of symbols based on a panel of "expert" before compressing the symbols by means of arithmetic coding. Statistical methods tend to be more computational intensive than substitutional methods. For a review on more substitutional, statistical, and other methods, such as transformation, table, and grammar-based methods, the reader is referred to [4].

Most of the existing substitutional and statistical methods can be generally categorized into horizontal and vertical modes [4], [18]. In horizontal mode, sequences are compressed individually using the information contained in a single sequence. Horizontal mode is of interest for revealing statistical and structural properties of a single biological sequence. In contrast, vertical methods [4], [18] handle a set of DNA sequences as a whole and each sequence is compressed by referring to information contained in other sequences or the entire set. More sequences imply greater redundancy, compressibility, and also higher computational complexity. In vertical mode, homologous sequences are always collected to analyze the variances and evolutionary development of related organisms/species.

With the rapid development of the high-throughput whole genome sequencing technologies, the vertical genomic compression has surfaced to become a research hot spot in the field of genomics. The genomic compression is featured by compressing a set of new genome sequences or short reads with respect to an existing reference genome. For instance, [27]–[33] introduced the idea of aligning target genome sequences to a reference genome and then encode only the variances between the sequences and the reference genome for storage. The compression rates reported in [27]–[33] are significantly impressive and the genomic compression possesses great potential as one of the leading technologies in this field. However, it is noted that currently the genomic compression works only on resequencing data to which a complete reference genome is available. Most of the existing genomic compression methods also rely heavily on the existences of meta-data such as single nucleotide polymorphism map and large structural variations [32]. Nevertheless, accurate meta-data and reference genomes are not available in *de novo* sequencing or the fields of metagenomics and epigenomics [34]. For this reason, the traditional substitutional and statistical methods, which have no demand for reference data, will continue to play a crucial role in advancing the knowledge of genetic variation.

Different DNA-specific compression techniques are superior in their unique aspects. The cooperation of techniques could lead to more efficient compressions. This forms the core motivation of the current study, which introduces substitutional methods into the context of vertical mode. Currently, most of the substitutional methods are designed to compress sequence individually and handle only one or two types of DNA subsequence repeats in either exact or approximate matching. In contrast to the existing efforts, we consider here up to four repeat patterns, namely, direct, mirror, pairing, and inverted repeats (defined in Section III) with approximate matching and compress multiple sequences simultaneously using EA-based substitutional method. This paper is the first to propose using MA for optimizing DNA sequence compression codebook, so that the most commonly repeated fragments in sequences of interest can be identified and replaced by short code vector indexes.

B. Memetic Algorithm

MA [16] represents the earliest and simplest instantiation of memetic computing [35] and is one of the recent core growing research areas of computational intelligence. Inspired by Darwin's evolutionary theory and Dawkins' notion of meme, MA is generally viewed as being close to a form of population-based hybrid global evolutionary algorithm (EA) coupled with a learning procedure capable of local refinements. In diverse contexts, MAs are also commonly known as hybrid EAs, Baldwinian EAs, Lamarckian EAs, cultural algorithms, and genetic local search. Recent studies on MAs have revealed their success on a wide variety of real-world problems, such as flow shop scheduling [36], [37], capacitated arc routing [38], very large-scale integrated floorplanning [39], material structure prediction [40], financial portfolio optimization [41], [42], robot control or mission management [43], [44], image segmentation [45], e-learning knowledge exploration [46], and feature/gene selection [47]–[50]. Significant studies to explore the purpose, definitions, design guidelines, and architecture for effective memetic computing have recently been reported [51]–[54]. MAs have been shown to not only converge at higher quality solutions but also search more efficiently than their conventional counterparts [55]–[60].

To date, many genetic algorithm, genetic programming, evolution strategies, and differential evolution-based MAs have been proposed [61]. Several PSO-based MAs, although fewer in comparison, have also been introduced to overcome the drawbacks and improve the performance of conventional PSO. Vesterstrøm *et al.* [62] introduced the division-of-labor specialization inspired by social insects to improve the local exploitative capabilities of PSO. Poli and Stephens [63] hybridized a hill-climber with PSO for continuous optimization, where the physics of mass and forces are employed to guide exploration of the fitness landscapes. Liu *et al.* [36] proposed a PSO-based MA (PSOMA) for solving the permutation flow shop scheduling problem. Particularly, the Nawaz-Enscore-Ham_1 (NEH_1), simulated annealing-based, and pairwise-based local search methods are designed for incorporation into the PSOMA. Petalas *et al.* [64] proposed a memetic particle swarm optimization (MPSO) as a synergy of the conventional PSO and a local search based on the random walk with directional exploitation. Different schemes of applying local search on the individuals are investigated and the convergence of MPSO on unimodal problem is subsequently analyzed [64]. In multiobjective optimization, Liu *et al.* [65] designed a PSO-based MA, known as FMOPSO, which is featured with fuzzy global best models and a synchronous particle local search. The newly features introduced in FMOPSO are shown to assist in avoiding undesirable premature convergence, thus leading to a well-distributed Pareto front attained. Ono and Nakayama [66] also proposed an MA based on a multiobjective PSO and the quasi-Newton local search for robust optimization.

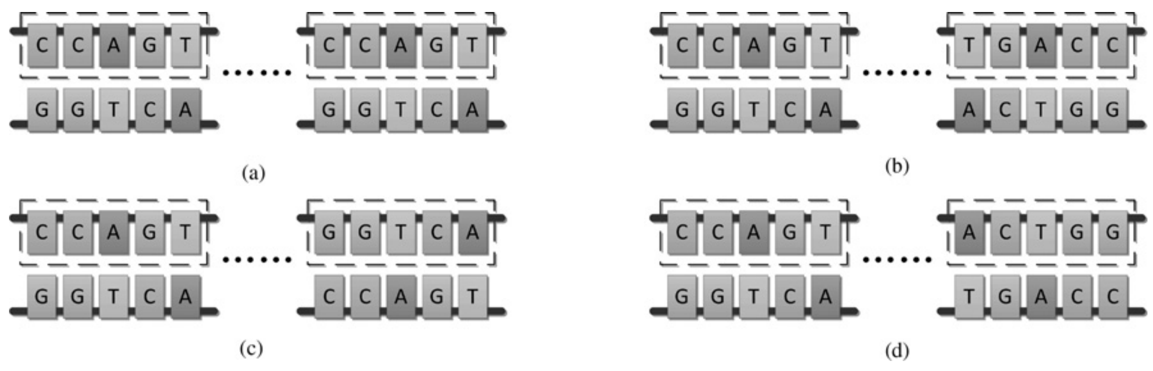


Fig. 2. Repeat patterns of DNA sequences. (a) Direct repeat. (b) Mirror repeat. (c) Pairing repeat. (d) Inverted repeat.

Despite the vast success of PSO-based MAs, problems and opportunities are continually emerging as intriguing challenges for the field. Domain knowledge and specially designed local search methods are still essential for the success of MAs on real-world problems. The balance between global and local search remains the main concern of designing efficient MA. In this paper, we propose a novel adaptive PSO-based MA, namely, POMA, incorporating a new ARV model for DNA sequence compression. The optimal efficiency of POMA is achieved by imposing an adaptive mechanism on AdpISPO-based local search. Particularly, POMA is capable of autonomously deciding the granularity of local search and selecting candidate solutions to undergo local search.

III. Repeat Patterns and ARV

In this section, we introduce the DNA repeat patterns and ARV model. DNA subsequence repeats always appear in patterns due to the peculiar biological characteristics. Taking the complementary double DNA strands shown in Fig. 2 as an example, there exist four repeat patterns, namely, direct, mirror, pairing, and inverted repeats [67] (similar definitions are also used in motif-finding [68]). The direct repeat is straightforward and the mirror repeat represents a reversed version of the original fragment. The pairing and the inverted repeats are constructed according to the base-pairing rules of A-T and G-C. As shown in Fig. 2(c), the pairing repeat "GGTCA" is the complement of the original fragment "CCAGT" based on the base-pairing rules. The inverted repeat "ACTGG" is shown in Fig. 2(d) as a reverse complement of "CCAGT." Among the four patterns, direct repeat is the main object of most DNA-specific compression algorithms. In this paper, all the four repeat patterns are considered as compression targets. Another issue of identifying DNA repeats is approximate matching. DNA repeats could be blurred due to base variations or sequencing errors. For example, the fragments "CCAGT" and "CCCGT" are approximate repeats with one base variation in the third position.

To denote the four repeat patterns with approximate matching, we propose a novel ARV model as follows. Given a subsequence referred as v, its direct, mirror, pairing, and inverted repeats are denoted as v, v^{-1}, v^{*}, and v^{-1*}, respectively. The symbols $^{-1}$ and * indicate the reversion and complementary operations on the original vector, respectively. ARV model also records base variations in approximate repeats in regard

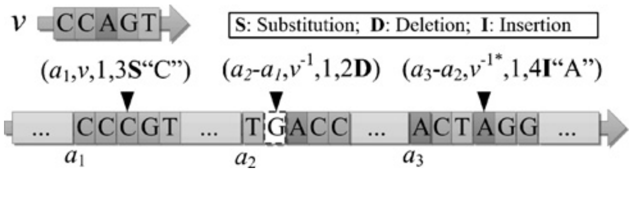

v CCAGT
S: Substitution; D: Deletion; I: Insertion
$(a_1,v,1,3\mathbf{S}$"C")
$(a_2\text{-}a_1,v^{-1},1,2\mathbf{D})$
$(a_3\text{-}a_2,v^{-1*},1,4\mathbf{I}$"A")
... CCCGT ... TGACC ... ACTAGG ...
a_1
a_2
a_3

Fig. 3. Approximate matching examples of ARVs.

of the variation position, type, and value. The types of base variation include insertions (I), deletions (D), and substitutions (S). Based on the definition of repeat patterns and base variation, our ARV model represents an approximate repeated fragment in a sequence by encoding its relative address, repeat pattern, edit distance (the number of base variations), and based variations.

Examples of ARVs are illustrated in Fig. 3. For the sake of simplicity, all examples are designed to contain only one base variation, i.e., the edit distance is one. This is not necessary the case with real-world DNA sequence compressing problems. Given a reference fragment v = "CCAGT," the first approximate direct repeat "CCCGT," with one substitution of "C" for "A" on the third base, is identified in position a_1 of the target sequence. The repeated fragment is thus denoted as (a_1, v, 1, 3S"C"). Similarly, the second matched fragment "TACC" in position a_2 is denoted as ($a_2 - a_1$, v^{-1}, 1, 2-D), which reads an approximate mirror repeat of v with the second base "G" deleted. Note that here the relative address $a_2 - a_1$, rather than the absolute address a_2, is used to save on bits in encoding the fragment location. The third matched fragment "ACTAGG" in position a_3 is an approximate inverted repeat of v with an "A" inserted as the fourth base. The fragment is denoted as ($a_3 - a_2$, v^{-1*}, 1, 4I"A").

DNA sequence compression can be formulated as a combinatory optimization of ARV codebook design. Given a codebook of N reference code vectors with each having M bases, it takes $\lceil \log_2 \max_{ra} \rceil$, $\lceil \log_2 N \rceil$, $\lceil \log_2 \max_{ed} \rceil$, and 2 bits to store the relative address, code vector index, edit distance, and repeat pattern of a repeated fragment, respectively. It is worth highlighting that the maximal relative address $\max_{ra}$ and maximal edit distance $\max_{ed}$ of all repeated fragments must be applied to every single repeated fragment. Storing a base variation requites $\lceil \log_2 M \rceil + 4$ bits. Particularly, the variation position takes $\lceil \log_2 M \rceil$ bits, and the variation type and value each takes 2 bits. The optimal ARV codebook is characterized

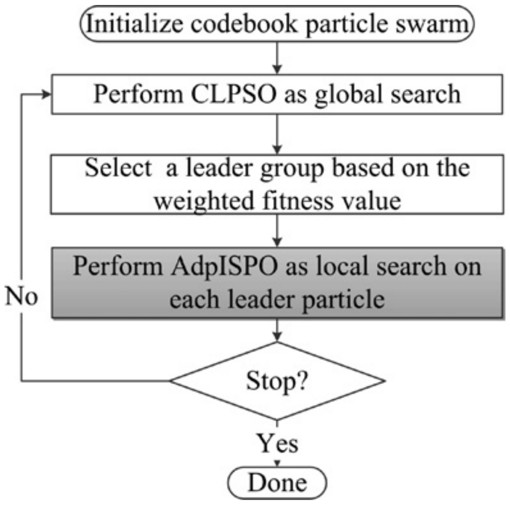

Initialize codebook particle swarm
Perform CLPSO as global search
Select a leader group based on the weighted fitness value
Perform AdpISPO as local search on each leader particle
No
Stop?
Yes
Done

Fig. 4. Flowchart of POMA.

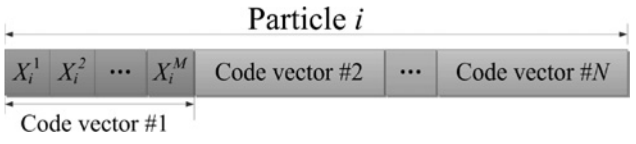


Fig. 5. Particle encoding.

with maximal cover rate of the repeats and minimal number of base variations. To solve this optimization problem, the adaptive POMA is proposed and described in the next section.

IV. POMA

In this section, we give details on the proposed POMA. The procedure of POMA is outlined in Fig. 4. In particular, candidate ARV codebooks are first encoded as particles and the CLPSO is used to perform global search. In each iteration of the global search, a leader group of particles is selected from the swarm based on weighted fitness values and the AdpISPO-based local search is then applied on each leader particle to refine the solution. The detailed descriptions of the particle encoding, fitness function, CLPSO-based global search, and AdpISPO-based local search are provided as follows.

A. Particle Encoding and Fitness Function

To encode the ARV codebook, the structure of particles in POMA is defined as Fig. 5.

A particle is formed by concatenating all code vectors in the codebook end to end. The size and the number of the code vectors in the codebook are configured as M and N, respectively, so the number of dimensions of each particle is MN. In conventional PSO, the particle position is denoted in the form of continuous values. In contrast the elements of the codebook are discrete symbols. Hence, a value mapping must be carried out before fitness evaluation as follows:

$$\Theta_i^j = \begin{cases} A, & if\ -\hat{X} \leq X_i^j < -\hat{X}/2 \\ G, & if\ -\hat{X}/2 \leq X_i^j < 0 \\ C, & if\ 0 \leq X_i^j < \hat{X}/2 \\ T, & if\ \hat{X}/2 \leq X_i^j \leq \hat{X} \end{cases} \tag{1}$$

where each dimension X_i^j is mapped to a base symbol $\Theta_i^j \in \{A, C, G, T\}$. $\hat{X}(> 0)$ and $-\hat{X}$ set the upper and lower bound for X_i^j, respectively. The effects of the value settings for $\hat{X}$, M, and N are investigated in Sections V-B2 and V-B1.

Here we formulate the optimization of codebook design as a maximization problem. The reciprocal of BPB is introduced as the fitness value for evaluating the quality of a particle as follows:

$$\text{Fitness}(X) = \frac{\#bases}{\#bits} \tag{2}$$

where $\#bases$ is the total number of bases in all sequences and $\#bits$ is the number of bits used to store the sequences. $\#bits$ is defined as:

$$\begin{aligned} \#bits &= 2MN + 2(\#base - \text{Covers}(X \to \Theta)) \\ &\quad + \eta\text{Repeats}(X \to \Theta) + \zeta\text{Variations}(X \to \Theta) \end{aligned} \tag{3}$$

where the following holds true.

1) $2MN$ is the number of bits used to store the codebook of size MN, i.e., naive 2 bits for each base.
2) $2(\#base - \text{Covers}(X \to \Theta))$ is the number of bits used to store the non-repeated bases. Here, $\text{Covers}(X \to \Theta)$ counts the total number of bases contained in all repeated fragments.
3) $\eta\text{Repeats}(X \to \Theta)$ denotes the number of bits for storing all repeated fragments excluding the base variations. In particular, $\text{Repeats}(X \to \Theta)$ denotes the total number of repeated fragments covered by the codebook and $\eta = \lceil \log_2 \max_{ra} \rceil + \lceil \log_2 N \rceil + \lceil \log_2 \max_{ed} \rceil + 2$ represents the number of bits used to store a single repeated fragment (as illustrated in Fig. 3) without considering the base variations.
4) $\zeta\text{Variations}(X \to \Theta)$ indicates the number of bits used to encode the based variations in all repeated fragments. In particular, $\text{Variations}(X \to \Theta)$ calculates the total number of bases variations and $\zeta = \lceil \log_2 M \rceil + 4$ denotes the number of bits for storing a single base variation.

In this paper, the fast fuzzy string searching algorithm AGREP [69] is used to identify the approximate repeats of ARVs in all concerned sequences, so that $\text{Covers}(X \to \Theta)$, $\text{Repeats}(X \to \Theta)$, and $\text{Variations}(X \to \Theta)$ can be determined. The process of the fitness evaluation is outlined in Fig. 6.

B. Global Search

To search the optimal ARV codebook, a global search is first performed based on CLPSO [5], [6]. CLPSO represents one of the PSO variants that attempts to mitigate the issue of premature convergence. It updates the velocity $V_i = (V_i^1, V_i^2, \ldots, V_i^d, \ldots, V_i^D)$ and the position $X_i = (X_i^1, X_i^2, \ldots, X_i^d, \ldots, X_i^D)$ of the ith particle based on

$$V_i^d = wV_i^d + cr_i^d(pbest(f_i^d)^d - X_i^d) \tag{4}$$

$$X_i^d = X_i^d + V_i^d \tag{5}$$

where d denotes the index of dimension, w is the inertia weight proposed by Shi and Eberhart [8] to balance the exploration and exploitation on the search space, c is an acceleration

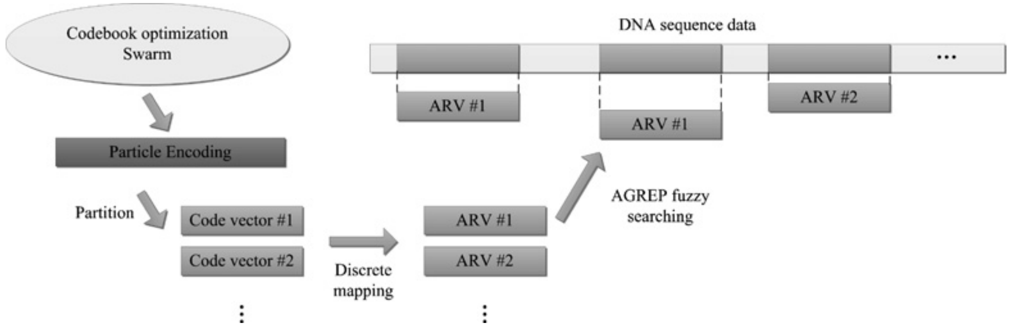


Fig. 6. Process of the fitness FE.

constant, r_i^d is a random number in [0, 1], and $pbest(f_i^d)^d$ represents the dth dimension of the exemplar particle's $pbest$. The index f_i^d of the exemplar particle is decided as follows.

1) For each dimension, a random number in [0, 1] is generated. If this random number is larger than a predefined learning probability Pc_i, then $f_i^d = i$; otherwise, two particles (excluding particle i) are randomly selected from the swarm using tournament selection and f_i^d is assigned the one with a better fitness. If $\forall d \in \{1, 2, \ldots, D\}$, $f_i^d = i$, a randomly selected dimension is forced to learn from another particle's $pbest$.
2) Once the exemplars for all dimensions of a particle are selected, the particle keeps learning from these exemplars until a stagnation of m (called refreshing gap) generations is detected. The number of stagnation generations of each particle is traced with a flag labeled as $flag$. Once $flag \geq m$, f_i^d is renewed based on 1) and $flag$ is reset.

Unlike the conventional PSO which updates all dimensions of a particle according to its own best previous position $pbest(i)$ and the global best position $gbest$, CLPSO enables each dimension of a particle to learn from any counterpart particle's $pbest$.

Using the novel comprehensive learning strategy whereby all other particles' historical best information is used to update a particle's velocity, CLPSO largely increases the search space of each particle so that the effect of premature convergence can be mitigated effectively. The global optimization capability of CLPSO is remarkable, but it still lacks for exploitation capability in local area. To overcome this drawback, an AdpISPO-based local search is introduced to the CLPSO swarm in each generation. The details of the resulting POMA are shown in Algorithm 1.

As shown in Algorithm 1, particles are generated randomly in initialization and a punishment factor $h_i \in [0, 1]$, which is used in local search, is also assigned to each particle. Each X_i, V_i, and the key parameters w, m, and $flag$ are updated iteratively following [6]. In each iteration, the best l_g particles in terms of the weighted fitness fw_i is selected to form a leader group. The local refinement is then applied to each particle in this group. The weighted fitness fw_i is determined as follows:

$$fw_i = \text{Fitness}(pbset(i)) - h_i \Delta Fitness_i \tag{6}$$

Algorithm 1 Procedure of POMA

```
1:  BEGIN
2:  Initialize a particle swarm $ps$, $gbest$, $pbest$, and the
    punishment factor $h_i = 0$, for $i = 1, 2, \ldots, |ps|$;
3:  Set $c$, $Pc_i$, $m$, and $flag$ following [6];
4:  while stopping criterion is not satisfied do
5:      Renew $w$ and $f_i^d$ following [6];
6:      Update each $V_i$ and $X_i$ based on (4) and (5), respectively;
7:      Calculate the fitness of each particle based on (2);
8:      Update $pbest$, $gbest$, and $flag$;
9:      Calculate the weighted fitness value $fw_i$ of each particle
        according to (6);
10:     Select a leader group of size $l_g$ based on $fw_i$;
11:     for each particle $ps(j)$ in the leader group do
12:         $Last_{best}$ = Fitness($pbest(j)$);
13:         Perform AdpISPO-Based Local Search on $ps(j)$;
14:         Evaluate the fitness improvement $\delta$ =
            (Fitness($pbest(j)$) $- Last_{best})/Last_{best}$;
15:         $h_j = min(1, h_j + 0.1)$ if $\delta < 0.001$;
16:     end for
17: end while
18: END
```

where

$$\Delta Fitness_i = \text{Fitness}(pbset(i)) - \min_{j=1}^{|ps|}(\text{Fitness}(pbest(j))).$$

Particles with greater $pbest$ will more likely join the leader group and undergo local search. However, if a leader particle has reached a local optimum, further local exploitation on the particle should not proceed. The punishment factor h_i is thus introduced to prevent the stagnant particles from the poor usage of computational resources. If the fitness of a leader particle is not improved through the local search, the corresponding punishment factor increases by 0.1 each time until the upper bound of 1 is reached. From (6), it is notable that increasing h_i will reduce fw_i, which is proportioned to the possibility of particle i to enter the leader group. The introduction of h_i thus allows more particles to undergo the AdpISPO-based local refinement in the discovery of multiple local optima.

Algorithm 2 Procedure of ISPO

```
1:  INPUT: a particle $ps(i)$;
2:  BEGIN
3:  Calculated Fitness($X_i$);
4:  for $k = 1$ to $K_I$ do
5:      for $d = 1$ to $D$ do
6:          $L_i^d = 0$;
7:          Update $V_i^d$, $X_i^d$, and $L_i^d$ according to (7), (8), and (9),
            respectively, for $J$ times;
8:      end for
9:  end for
10: END
```

TABLE I
PARAMETER SETTINGS OF THE ALGORITHMS

Algorithm	Parameters	Reference
PSOw	$\|ps\| = 20$, $w = 5$, $c1 = c2 = 2$	[5]
CLPSO	$\|ps\| = 10$, $w : 0.9 \sim 0.4$, $c = 1.49445$, $m = 8$	[6]
ISPO	$J = 30$, $a = 150$, $p = 10$	[70]
AdpISPO	$\|ps_l\| = 10$, $K_I = 10$, $J = 30$, $w : 0.9 \sim 0.4$, $c = 1.49445$	
POMA	$\|ps\| = 30$, $w : 0.9 \sim 0.4$, $c = 1.49445$, $m = 8$	
	$\|ps_l\| = 5$, $K_l = 5$, $K_I = 5$, $J = 10$, $L = 5$	

C. AdpISPO-Based Local Search

The AdpISPO is proposed based on the intelligent single particle optimizer (ISPO) [70], which searches with only one particle. Given a particle $ps(i)$, ISPO updates each dimension of the particle independently as follows:

$$V_i^d = a/j^p r + L_i^d \tag{7}$$

$$X_i^d = \begin{cases} X_i^d + V_i^d, & \text{if fitness is improved} \\ X_i^d, & \text{otherwise} \end{cases} \tag{8}$$

$$L_i^d = \begin{cases} 2V_i^d, & \text{if fitness is improved} \\ L_i^d/2, & \text{otherwise} \end{cases} \tag{9}$$

where the dth dimension of the particle's velocity V_i^d is updated with a diversity portion $a/j^p r$ and an inertia learning portion L_i^d. In (7), a denotes an accelerate factor for the diversity portion, j represents the iteration number, p is a descend factor to control the decay of the velocity, and r is a random number in $[-0.5, 0.5]$. The portion a/j^p, acting as a decreasing function of time, makes the particle explore more at the beginning and evolves toward exploitation as the search proceeds. Hence, the search region shrinks as ISPO proceeds, while the search intensity on unit area increases. From (8), it is notable that X_i^d is updated to $X_i^d + V_i^d$ only if the movement results in an improvement of the fitness, otherwise X_i^d remains unchanged. The learning factor L_i^d increases to $2V_i^d$ if the particle evolves with improvement in fitness, otherwise L_i^d is halved and the diversity portion kicks in to explore in new regions. The particle is updated for K_I iterations and each dimension is updated J times. The procedure of ISPO is outlined in Algorithm 2.

ISPO is a highly efficient local optimization approach with search behavior defined by step factor a and descend factor p. Since no priori knowledge is available in advance on suitable settings of these two parameters, an adaptive ISPO

Algorithm 3 Procedure of AdpISPO

```
1:  INPUT: a particle $ps(i)$;
2:  BEGIN
3:  Randomly generate a particle swarm $ps_l$ with each particle
    encoding a pair settings of $a$ and $p$;
4:  for $k = 1$ to $K_l$ do
5:      for $j = 1$ to $|ps_l|$ do
6:          Perform ISPO on $ps(i)$ based on the settings of $a$ and
            $p$ encoded in $ps_l(j)$;
7:          Calculate the fitness of $ps_l(j)$;
8:          Update the velocity and position of $ps_l(j)$ based on
            (4), (5);
9:      end for
10: end for
11: END
```

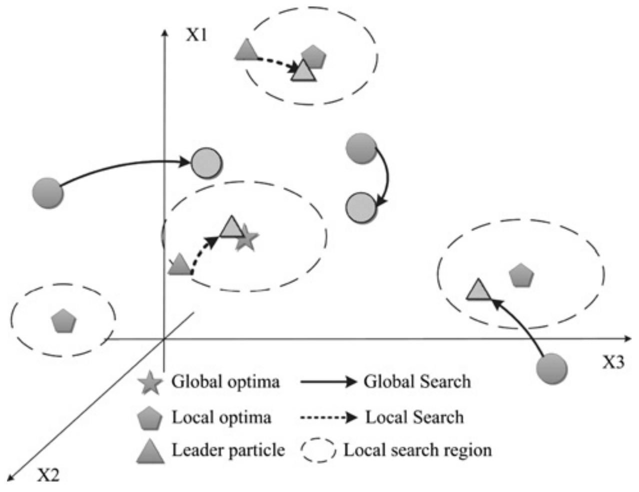


Fig. 7. Search process of POMA.

or AdpISPO in short, is proposed and described in what follows. Particularly, candidate pair settings of $a \in [-\hat{X}, \hat{X}]$ and $p \in [1, 60]$ are randomly generated and encoded in a particle swarm ps_l. The fitness of each pair settings of a and p is evaluated based on the improvement fitness of ISPO. Subsequently, the comprehensive learning strategy, i.e., updating rules defined in (4) and (5), is employed for adapting the values of a and p. The learning procedure is proceeded for K_l iterations. The procedure of the AdpISPO algorithm is shown in Algorithm 3.

By cooperating the CLPSO-based global search and AdpISPO-based local search, the search behavior of POMA in the solution space is illustrated in Fig. 7. Particularly, the populace particles normally explore the whole search space in global search, while some of the leader particles drill down to local optimal by undertaking local search.

V. RESULTS

This paper evaluates the performance of POMA on both benchmark functions and real DNA sequences. The standard PSOw [8], CLPSO, ISPO, and AdpISPO represent the counterpart algorithms identified for assessment against with POMA. PSOw acts as the baseline algorithm for all. Note that

TABLE II

PERFORMANCES OF THE ALGORITHMS OVER 100 RUNS ON THE 23 BASIC BENCHMARK FUNCTIONS

		PSOw	CLPSO	ISPO	AdpISPO	POMA
*f*1	Mean	3.2280E-37	1.5724E-07	**8.2467E-115**	1.2185E-19	1.0422E-20
	Var.	6.7313E-73	4.4717E-14	**4.3698E-230**	1.2174E-18	1.8128E-41
	Best	2.7423E-110	2.0333E-14	**4.1515E-115**	3.8750E-60	4.2629E-21
	Worst	3.7701E-36	7.8399E-07	**1.1260E-114**	1.2174E-17	1.6997E-20
*f*2	Mean	3.0000E+01	1.5161E-05	**3.8802E-58**	6.2513E-10	4.0861E-10
	Var.	2.8182E+02	2.5508E-10	**2.2781E-117**	6.1736E-09	8.9960E-21
	Best	1.0000E+01	1.3032E-09	**3.2333E-58**	2.8549E-30	2.4655E-10
	Worst	8.0000E+01	6.0891E-05	**4.7760E-58**	6.1740E-08	5.0022E-10
*f*3	Mean	1.9052E+04	1.1981E+04	**1.0776E-11**†	8.2379E+03	2.4359E+00
	Var.	5.5191E+07	2.0413E+07	**1.1748E-23**	2.7863E+03	1.5798E+00
	Best	5.0016E+03	4.1372E+03	**5.4618E-12**	2.9296E+03	2.4371E-02
	Worst	3.3333E+04	2.1126E+04	**1.6105E-11**	1.9382E+04	5.0169E+00
*f*4	Mean	2.0257E+00	2.1853E+01	**1.6311E-02**†	7.6512E+01	2.3133E+00
	Var.	5.8485E+00	8.6116E+01	**4.7119E-06**	2.0984E+01	4.4686E+00
	Best	9.6738E-03	1.1744E+01	1.2999E-02	2.1898E+01	**1.1017E-04**
	Worst	7.8801E+00	6.3684E+01	**2.0077E-02**	9.7118E+01	6.4208E+00
*f*5	Mean	2.0231E+05	1.1902E+02	8.7572E+00	5.1550E+01	**3.2024E-01**†
	Var.	1.6272E+11	6.9772E+03	5.7664E+01	5.8863E+01	**2.1744E-01**
	Best	2.2255E+00	1.7348E+01	**8.9313E-04**	1.7711E-03	1.5274E-02
	Worst	1.0001E+06	3.2488E+02	1.7077E+01	2.2811E+02	**1.2308E+00**
*f*6	Mean	1.9978E+03	9.0391E-08	9.5070E-21	**0.0000E+00**†	7.9917E-21
	Var.	1.6330E+07	1.4280E-14	1.7215E-41	**0.0000E+00**	1.9626E-41
	Best	**0.0000E+00**	2.4484E-14	3.9717E-21	**0.0000E+00**	4.0498E-21
	Worst	1.0100E+04	4.3278E-07	1.6272E-20	**0.0000E+00**	1.6278E-20
*f*7	Mean	8.3513E-01	4.7449E-18	2.9905E-25	1.8728E+01	**2.1754E-39**†
	Var.	2.5620E+00	7.4600E-35	1.1331E-49	1.2022E+01	**1.5066E-77**
	Best	**7.5438E-154**	8.5830E-29	4.3288E-26	1.0826E+01	1.4245E-40
	Worst	5.3687E+00	3.6550E-17	1.2689E-24	8.2210E+01	**9.8482E-39**
*f*8	Mean	3.6314E+03	5.9123E+02	**1.3382E-02**†	1.8748E+03	7.3433E+02
	Var.	3.3626E+05	2.4119E+05	**1.5770E-19**	8.9484E+02	1.9511E+04
	Best	1.9011E+03	**1.3382E-02**	**1.3382E-02**	3.2730E+02	5.9221E+02
	Worst	4.7764E+03	1.9011E+03	**1.3382E-02**	4.8618E+03	9.4752E+02
*f*9	Mean	1.2375E+02	2.3218E+00	**0.0000E+00**	2.9239E+01	**0.0000E+00**
	Var.	1.6086E+03	8.1489E+01	**0.0000E+00**	3.1606E+01	**0.0000E+00**
	Best	4.8753E+01	4.7727E-10	**0.0000E+00**	3.7290E-07	**0.0000E+00**
	Worst	1.9025E+02	5.0743E+01	**0.0000E+00**	1.5007E+02	**0.0000E+00**
*f*10	Mean	5.8902E+00	**4.1488E-01**†	4.1570E-01	1.0126E+01	4.1570E-01
	Var.	6.1831E+01	8.5368E-07	0.0000E+00	7.3009E+00	**0.0000E+00**
	Best	4.1570E-01	4.1247E-01	4.1570E-01	**1.3677E-07**	4.1570E-01
	Worst	1.9547E+01	**4.1570E-01**	**4.1570E-01**	1.9829E+01	**4.1570E-01**
*f*11	Mean	1.6029E+01	**4.1534E-06**†	2.1115E-02	5.5237E-02	1.8079E-03
	Var.	1.5832E+03	**3.6056E-11**	8.7032E-04	7.6256E-02	1.0332E-05
	Best	**0.0000E+00**	1.8121E-12	7.2211E-06	**0.0000E+00**	**0.0000E+00**
	Worst	1.8030E+02	**2.7372E-05**	9.2913E-02	4.9450E-01	7.3960E-03
*f*12	Mean	2.2526E-31	**7.9555E-33**	1.9009E-31	1.1152E-03	1.7884E-23
	Var.	2.6970E-62	**9.2273E-64**	2.1800E-62	1.0824E-02	4.4743E-46
	Best	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	3.0166E-17	3.8957E-24
	Worst	4.3330E-31	**1.4135E-31**	4.3330E-31	1.0823E-01	5.9411E-23
*f*13	Mean	**2.6154E-16**	1.4689E-06	**2.6154E-16**	2.0980E-02	2.6156E-16
	Var.	**0.0000E+00**	3.9940E-12	**0.0000E+00**	1.6241E-01	2.8124E-41
	Best	2.6154E-16	1.8467E-13	2.6154E-16	**2.8774E-17**	2.6155E-16
	Worst	2.6154E-16	8.5019E-06	**2.6154E-16**	1.6084E+00	2.6156E-16
*f*14	Mean	1.1111E-01	**1.5484E-07**	**1.5484E-07**	9.0535E-02	**1.5484E-07**
	Var.	2.3737E-01	**0.0000E+00**	**0.0000E+00**	5.3399E-01	**0.0000E+00**
	Best	**1.5484E-07**	**1.5484E-07**	**1.5484E-07**	1.9962E-03	**1.5484E-07**
	Worst	3.0000E+00	**1.5484E-07**	**1.5484E-07**	4.9288E+00	**1.5484E-07**
*f*15	Mean	5.7026E-03	6.0675E-04	3.8034E-02	5.3384E-04	**2.2867E-04**†
	Var.	7.0780E-05	1.2142E-06	2.3613E-03	2.9178E-04	**6.6468E-09**
	Best	1.4012E-08	1.4899E-04	**1.1878E-08**	2.0405E-08	7.1602E-05
	Worst	2.0056E-02	7.7305E-03	1.2257E-01	1.2886E-03	**2.8849E-04**

		PSOw	CLPSO	ISPO	AdpISPO	POMA
*f*16	Mean	**4.6510E-08**	**4.6510E-08**	4.7621E-08	**4.6510E-08**	**4.6510E-08**
	Var.	**0.0000E+00**	**0.0000E+00**	5.5556E-17	2.4653E-13	**0.0000E+00**
	Best	4.6510E-08	4.6510E-08	4.6510E-08	4.6510E-08	4.6510E-08
	Worst	**4.6510E-08**	**4.6510E-08**	9.6510E-08	4.6513E-08	**4.6510E-08**
*f*17	Mean	2.5648E-01	**1.1264E-04**	4.6157E-01	**1.1264E-04**	**1.1264E-04**
	Var.	5.3773E-01	**7.2190E-24**	8.7112E-01	3.8763E-12	**7.2190E-24**
	Best	1.1264E-04	1.1264E-04	1.1264E-04	1.1264E-04	1.1264E-04
	Worst	2.3074E+00	**1.1264E-04**	2.3074E+00	**1.1264E-04**	**1.1264E-04**
*f*18	Mean	**0.0000E+00**	**0.0000E+00**	1.9200E+01	7.8511E-01	**0.0000E+00**
	Var.	**0.0000E+00**	**0.0000E+00**	6.5022E+02	4.3431E+00	**0.0000E+00**
	Best	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	8.0824E-14	**0.0000E+00**
	Worst	**0.0000E+00**	**0.0000E+00**	8.1000E+01	2.7028E+01	**0.0000E+00**
*f*19	Mean	**2.7821E-03**	**2.7821E-03**	1.6380E+00	**2.7821E-03**	**2.7821E-03**
	Var.	**0.0000E+00**	**0.0000E+00**	1.5322E+00	5.8061E-08	**0.0000E+00**
	Best	2.7821E-03	2.7821E-03	2.7821E-03	**2.7817E-03**	2.7821E-03
	Worst	**2.7821E-03**	**2.7821E-03**	2.8592E+00	**2.7821E-03**	**2.7821E-03**
*f*20	Mean	7.7855E-02	1.7316E-02	5.0510E-02	**1.9938E-03**	1.9952E-03
	Var.	1.0263E-02	1.5603E-03	3.2940E-03	1.3605E-05	**4.3122E-21**
	Best	1.9952E-03	1.9948E-03	1.9952E-03	**1.8591E-03**	1.9952E-03
	Worst	3.6358E-01	1.1690E-01	1.1690E-01	**1.9952E-03**	**1.9952E-03**
*f*21	Mean	3.9094E+00	1.2089E+00	4.2458E+00	**4.0298E-01**	1.2079E+00
	Var.	8.1443E+00	3.9894E+00	9.5767E+00	**1.5887E+00**	3.9817E+00
	Best	1.5320E-01	**1.5289E-01**	1.5320E-01	1.5320E-01	1.5320E-01
	Worst	7.3695E+00	4.9448E+00	7.3695E+00	**4.8992E+00**	**4.8992E+00**
*f*22	Mean	3.3748E+00	1.5603E+00	5.0762E+00	**6.9606E-01**	1.3959E+00
	Var.	8.6033E+00	4.3066E+00	7.1524E+00	**2.2195E+00**	3.5292E+00
	Best	**4.0294E-01**	**4.0294E-01**	**4.0294E-01**	**4.0294E-01**	**4.0294E-01**
	Worst	7.2481E+00	6.2757E+00	7.2481E+00	6.2757E+00	**4.8712E+00**
*f*23	Mean	4.0652E+00	2.4270E+00	5.8005E+00	**1.2161E+00**†	3.9846E+00
	Var.	9.0912E+00	5.7855E+00	7.8205E+00	**2.5505E+00**	9.8550E+00
	Best	5.3641E-01	5.3641E-01	5.3641E-01	**4.8327E-01**	5.3641E-01
	Worst	7.5783E+00	7.5727E+00	8.1405E+00	**6.2085E+00**	7.1289E+00

CLPSO, ISPO, and AdpISPO denote the basic ingredients of the POMA. By pitting POMA against each of its components helps reveal the effect and contribution of each ingredients that makes up the proposed algorithm.

The parameters of all the algorithms are summarized in Table I. PSOw and CLPSO are configured as in [5] and [6], respectively, while ISPO according to [70]. For AdpISPO and POMA, the parameters w, c, and m are configured as in [6], while the other parameters including: 1) $|ps_l|$: the swarm size of the parameter particles; 2) K_I: the maximum generation of ISPO; 3) J: the maximum iteration for dimensional updating in ISPO; and 4) K_l: the maximum generation for parameter adaption, are empirically set to small values to avoid premature convergence in local optima. The size of the leader group of POMA, l_g, is configured based on the empirical study conducted in Section V-A2. All algorithms are given a maximum number of 2.5E+05 fitness function calls to ensure fair comparisons. As the computational overhead of the fitness evaluation is much greater than the other computational cost of evolutionary operations in PSOs, especially in the context of DNA sequence compression, the variant in computation cost incurred by different updating operations in the PSOs can be considered to be negligible.

A. Results on Benchmark Functions

To investigate the search behavior of the proposed POMA, 23 commonly used basic benchmark functions from [71] and 6 composite functions proposed by Liang *et al.* [5] are used. Note that POMA is applicable to continuous problem with conventional particle encoding scheme. The mean, variance, best, and worst fitness values obtained by all the algorithms over 100 runs, on the basic and composite benchmark functions are reported in Tables II and III, respectively. Statistical comparisons of the algorithms on all the functions are conducted using the two-tailed paired t-tests. Significant differences observed at level of $\alpha = 0.05$ are then highlighted with superscript † in the tables.

Table II shows that ISPO exhibits better overall performance than the other algorithms on the unimodal functions, i.e., $f1$–$f7$. Since ISPO is specially designed to exploit a local area more effectively than other algorithms, it is unsurprising for it to converge and locate the single optimal on unimodal problems. On multimodal functions ($f8$–$f23$) and the composite functions, on the other hand, ISPO is easily stuck in some local optima and hence often miss the global optimum solution. The results on ($f8$–$f23$) showed that CLPSO, AdpISPO, POMA attained better fitness quality on multimodal benchmark functions than ISPO and PSOw. This suggests that the specialized learning strategies used in CLPSO, AdpISPO, and POMA are effective in mitigating the effect of premature convergence and thus provide more chances in the discovery of the global optima. The performance differences between CLPSO, AdpISPO, and POMA on most of the multimodal functions ($f8$–$f23$) are not statistically significant.

The 23 basic benchmark functions are thus seen to be too simple for distinguishing any statistical performance differences among CLPSO, AdpISPO, and POMA. As a result, the comparison study is further extended to involve the composite functions ($cf1$–$cf6$) which are often regarded as more complex and difficult to solve. As shown in Table III, POMA, which

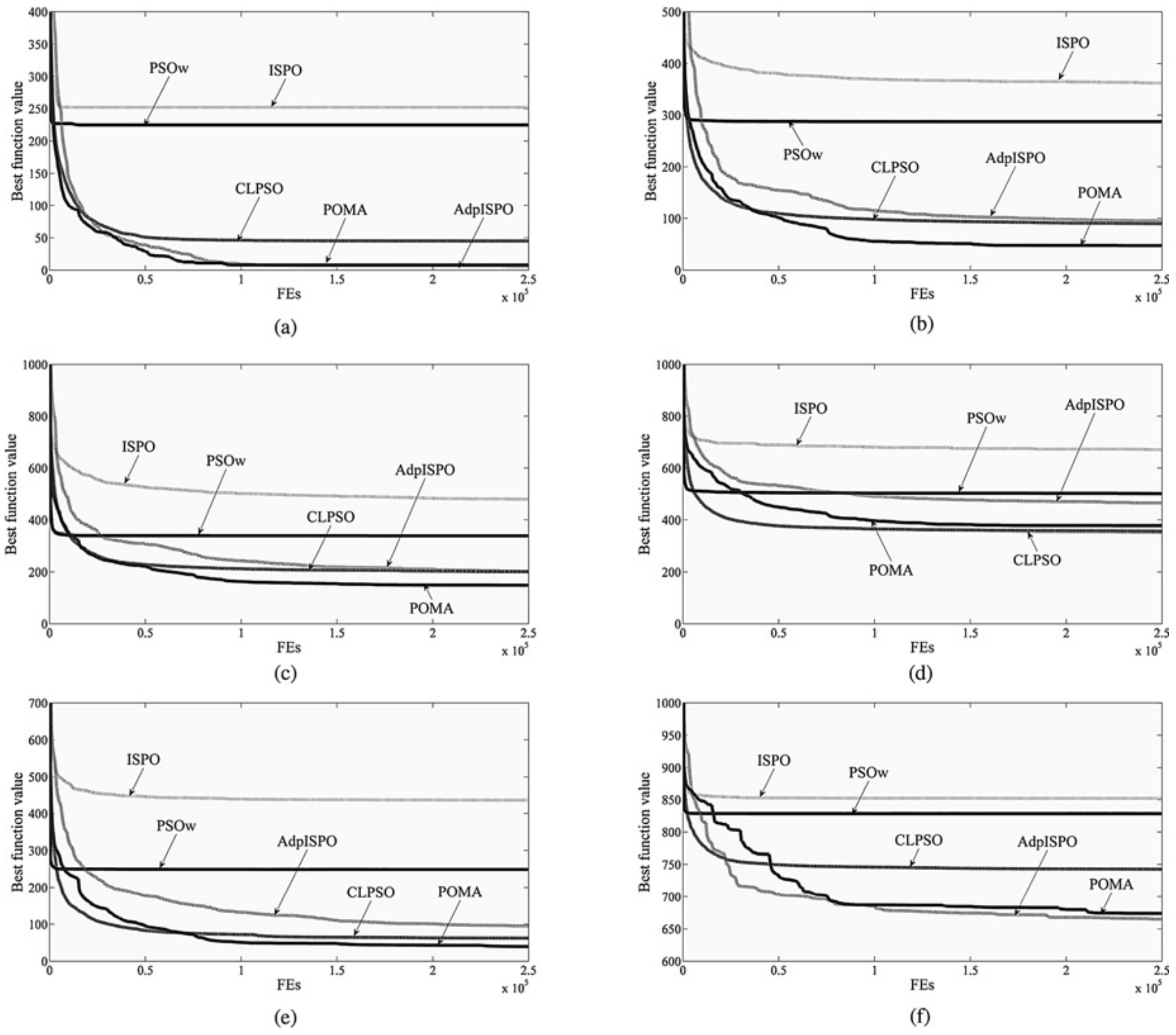


Fig. 8. Average convergence trace of the algorithms over 100 runs on the composite functions. (a) Composite function 1. (b) Composite function 2. (c) Composite function 3. (d) Composite function 4. (e) Composite function 5. (f) Composite function 6.

inherits both the strengths of the AdpISPO as local search and the CLPSO as global search, displayed the the overall best performance. Using statistical t-test, POMA is found to significantly outperform the CLPSO and AdpISPO on $cf2$, $cf3$, and $cf5$. On $cf1$, $cf4$, and $cf6$, POMA is also among the competitive algorithms that exhibit the best mean fitness values.

1) *Convergence Trace:* The average convergence traces of all algorithms over 100 runs on the composite functions are also plotted in Fig. 8. The figures show that POMA converges faster than most of the other algorithms in terms of function evaluations (FEs). In other words, POMA achieves better fitness quality at fewer FEs.

2) *Local Search Frequency:* The degree of global and local search are defined by the parameters listed in Table I. Among these parameters, there exists several rules of thumb in the literatures [6], [70] on the configurations of PSO pertaining to the swarm size and maximum number of iterations in CLPSO and ISPO. Here, we investigate the effect of local search frequency, which is defined as $l_g/|ps|$, on POMA. We fix $|ps|$=30 and study the performance of POMA with increasing l_g, i.e., increasing local search frequency, on the complex composite functions. The results shown in Fig. 9 suggest that POMA with zero or very low local search frequency of $1/|ps|$ did not lead to good search performance. As the local search frequency increases from 1/6 to 1, the search performance of POMA does not vary significantly on most of the composite functions, whereas POMA for a local search frequency of 1/6, i.e., $l_g = 5$, attained the best overall convergence rate. The observation suggests that POMA does not require intense local search to give promising performance.

B. Results on Benchmark DNA Sequences

Despite the encouraging results of POMA on benchmark functions, the efficacy of the proposed approach on real-world problem, i.e., the motivating problem for us here is DNA sequence compression, remains yet to be assessed. Here 11 most commonly used benchmark DNA sequences [72] are considered as a set for assessing the performances of the algorithms. The details of the DNA sequences are summarized in Table IV.

To the best of our knowledge, no evolutionary computation technique-based DNA compression algorithms have been explored in the literature, and the present serves as a novel

TABLE III

PERFORMANCES OF THE ALGORITHMS OVER 100 RUNS ON THE SIX COMPOSITE BENCHMARK FUNCTIONS

		PSOw	CLPSO	ISPO	AdpISPO	POMA
*cf*1	Mean	2.2480E+02	4.5070E+01	2.5200E+02	**6.0801E+00**	8.0000E+00
	Var.	1.2598E+02	7.7077E+01	2.0572E+02	**2.3859E+01**	2.7266E+01
	Best	**0.0000E+00**	**0.0000E+00**	1.5407E-33	**0.0000E+00**	**0.0000E+00**
	Worst	6.4069E+02	3.0000E+02	7.0000E+02	**1.0000E+02**	**1.0000E+02**
*cf*2	Mean	2.8735E+02	8.9864E+01	3.6201E+02	9.4955E+01	**4.7294E+01**†
	Var.	1.4374E+02	9.8233E+01	1.8438E+02	6.6496E+01	**5.9841E+01**
	Best	4.3375E+01	1.7968E+00	1.3577E+00	4.3949E-07	**0.0000E+00**
	Worst	6.9805E+02	4.1528E+02	8.6265E+02	3.0555E+02	**2.6316E+02**
*cf*3	Mean	3.4074E+02	2.0106E+02	4.8004E+02	2.0152E+02	**1.4877E+02**†
	Var.	1.3274E+02	1.4414E+02	2.6332E+02	6.3818E+01	**3.7756E+01**
	Best	1.3163E+02	**9.2352E+01**	1.1965E+02	1.1143E+02	9.5092E+01
	Worst	7.2958E+02	8.7452E+02	1.1837E+03	4.1010E+02	**3.1267E+02**
*cf*4	Mean	5.0107E+02	**3.5604E+02**	6.7123E+02	4.6542E+02	3.7785E+02
	Var.	1.7623E+02	9.3505E+01	1.5254E+02	8.3136E+01	**5.1540E+01**
	Best	3.0095E+02	**2.8484E+02**	3.6613E+02	3.0471E+02	2.8777E+02
	Worst	9.1035E+02	8.0000E+02	1.3554E+03	7.3435E+02	**5.9818E+02**
*cf*5	Mean	2.4901E+02	6.2485E+01	4.3599E+02	9.5275E+01	**3.9736E+01**†
	Var.	1.6932E+02	1.1565E+02	2.6442E+02	7.0246E+01	**5.2054E+01**
	Best	1.9741E+01	**9.2514E-01**	7.1785E+01	5.7701E+00	3.1143E+00
	Worst	6.9275E+02	5.5428E+02	1.0317E+03	2.8685E+02	**2.2721E+02**
*cf*6	Mean	8.2835E+02	7.4258E+02	8.5196E+02	**6.6481E+02**	6.7380E+02
	Var.	1.5038E+02	1.8077E+02	**1.2938E+02**	1.6014E+02	1.8140E+02
	Best	4.2308E+02	4.6600E+02	5.2184E+02	4.4872E+02	**4.1850E+02**
	Worst	9.2861E+02	9.0362E+02	9.3978E+02	9.0817E+02	**9.0349E+02**

TABLE IV

ELEVEN BENCHMARK DNA SEQUENCES

Sequence	Length (Bases)	Summary
CHMPXX	121 024	Marchantia polymorpha chloroplast genome DNA
CHNTXX	155 844	Nicotiana tabacum chloroplast genome DNA
HEHCMVCG	229 354	Human cytomegalovirus strain AD169 complete genome
HUMDYSTROP	38 770	Homo sapiens dystrophin (DMD) gene, intron 44
HUMGHCSA	66 495	Human growth hormone (GH-1 and GH-2) and chorionic somatomammotropin (CS-1, CS-2, and CS-5) genes, complete cds
HUMHPRTB	56 737	Human hypoxanthine phosphoribosyltransferase (HPRT) gene, complete cds
HUMHDABCD	58 864	Human DNA sequence of contig comprising three cosmids (HDAB, HDAC, HDAD)
HUMHBB	73 308	Human beta globin region on chromosome 11
MPOMTCG	186 609	Marchantia polymorpha mitochondrion, complete genome
SCCHRIII	316 613	*S. cerevisiae* chromosome III complete DNA sequence
VACCG	194 711	Vaccinia virus, complete genome
Total	1 498 329	11.43 MB (8 bits per base)[a]

[a]Sequences normally are stored in ASCII characters in database.

attempt to comprehensively study the use of PSOs and MAs for DNA sequence compression. Besides PSOw, CLPSO, ISPO, AdpISPO, and POMA, two state-of-the-art gradient-based local search strategies, namely, Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG) [73] and Davidon, Fletcher, and Powell Strategy (DFP) [74] are also considered. Both DSCG and DFP have been shown to be very efficient local improvement procedures even on multimodal problems [53], [75]. Hence, DSCG and DFP are hybridized with CLPSO to form two other memetic PSOs, which are denoted as CLPSO-DSCG and CLPSO-DFP, respectively. All the PSOs and MAs use consistent parameter settings of $M = 8$, $N = 32$, and $\hat{X} = 10$ for particle encoding. The mean best fitness values of the algorithms across 30 independent runs on all 11 DNA sequences are then reported in Table V. The paired two-tailed t-test result shows that POMA attained significant better BPB than all the other counterpart algorithms. The convergence traces of the algorithms on the benchmark DNA data, as shown in Fig. 10, suggest POMA exhibits a faster convergence speed in terms of FEs. The observation is consistent with the results on the composite benchmark functions.

When designing new evolutionary, swarm, or memetic algorithms, besides pitting against algorithms of similar field, i.e., evolutionary computation, it is more important to assess the proposed algorithm against other existing state-of-the-arts for solving the same problem of interest. In DNA sequence data, the attained BPB is of particular importance when assessing the performance of any compression methods. Here the general-purpose text compression algorithm bzip2 and six state-of-the-art DNA-specific compression algorithms, including BioCompress2 [19], GenCompress [20], CTW+LZ [17], DNAcompress [21], GeNML [22], and XM [26], and the EA-based algorithms are assessed in terms of BPB. The mean BPB results achieved by all algorithms are then summarized in Fig. 11. POMA is shown to achieve the best BPB of 1.536

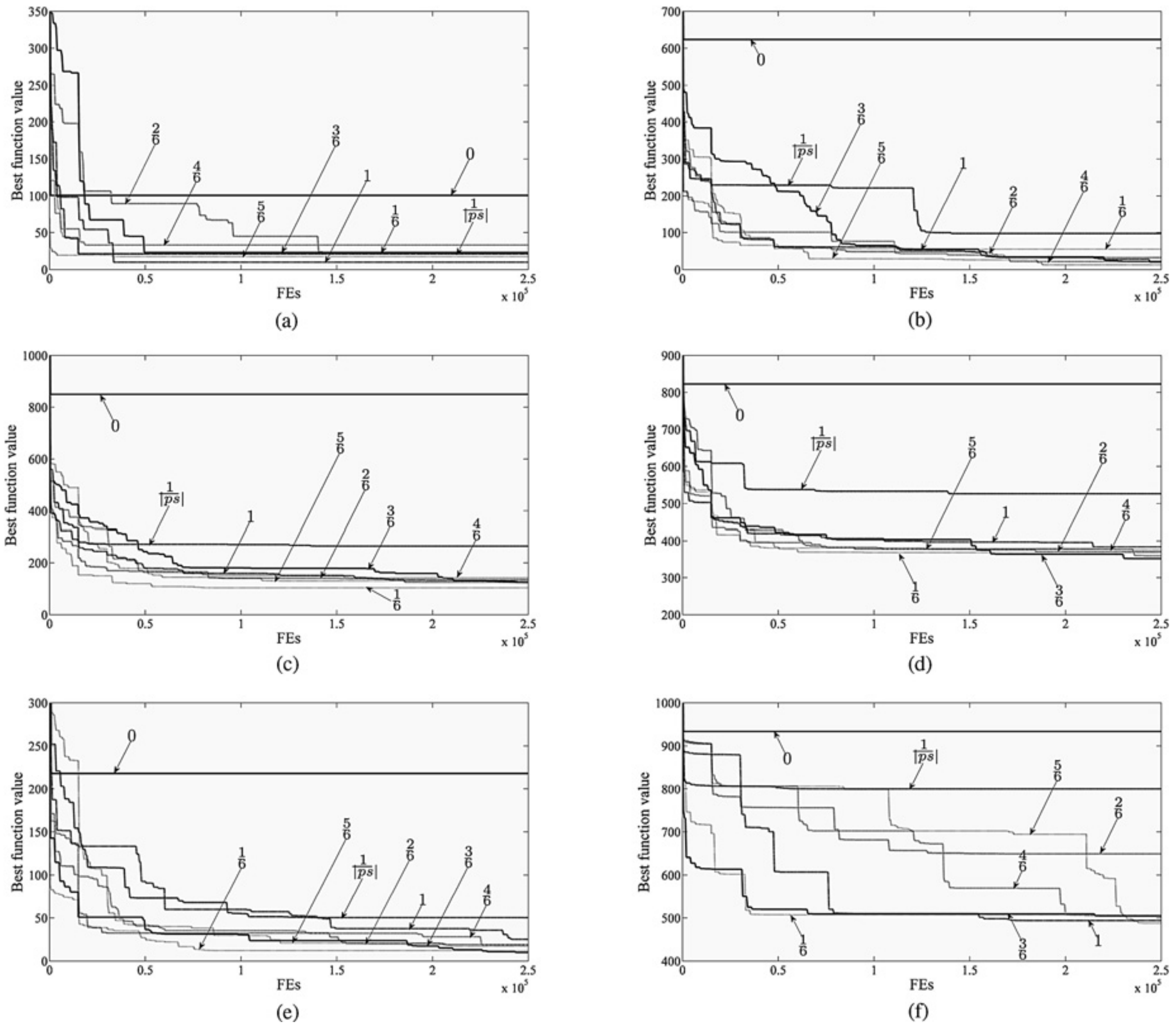


Fig. 9. Convergence traces of POMA for different local search frequencies on the composite functions. (a) Composite function 1. (b) Composite function 2. (c) Composite function 3. (d) Composite function 4. (e) Composite function 5. (f) Composite function 6.

among all the algorithms. Based on this BPB, the benchmark DNA data is compressed from the original size of 11.43 MB (ASCII, 8 BPB) to about 2.19 MB, i.e., the compression ratio is $\sim 5 : 1$. Using our proposed POMA, the BPB has been shown to have considerably improved over all the other algorithms. MAs are observed to have lower BPB than other algorithms. XM was recognized as one of the best single sequence compression algorithms in [4], [27], but it gives a BPB of only 1.694. The MAs, i.e., POMA, CLPSO-DFP, and CLPSO-DSCG, obtain better performances than DNA-specific compression algorithms and other intelligent search algorithms.

1) *Effects of M and N:* The solutions of possible codebook are defined by the size of the code vector M and the size of codebook N, i.e, the number of code vectors. Both M and N are configured to the power of two so as to avoid wasting bits for encoding the repeat fragments. The effects of M and N on the performance of POMA are investigated and the results are plotted in Fig. 12. When M increases from 8 to 128, i.e., 2^3 to 2^7, POMA exhibits best performance at $M = 8$. With $M = 2^2$ and 2^1, the sequences are non-compressible, i.e., the obtained BPBs are greater than two, so the corresponding results are not

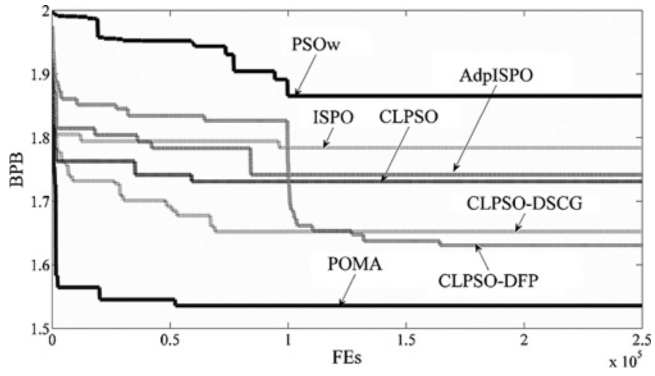


Fig. 10. Convergence traces of the algorithms on all 11 benchmark DNA sequences.

shown in Fig. 12. The BPB of POMA reduces for increasing N. However, BPB is observed to stabilize at $N = 32$. Any further rise of N does not improve BPB but only increases the space required to store the codebook. Hence, the rational configuration of N is 32.

TABLE V

PERFORMANCES OF THE ALGORITHMS ON BENCHMARK DNA SEQUENCES FOR 30 INDEPENDENT RUNS

	PSOw	CLPSO	ISPO	AdpISPO	POMA	CLPSO-DFP	CLPSO-DSCG
Mean	1.8654E+00	1.7309E+00	1.7839E+00	1.7417E+00	**1.5360E+00**†	1.6307E+00	1.6523E+00
Var.	7.1747E-03	5.1411E-04	9.4490E-04	2.9622E-03	**1.8832E-04**	2.6257E-03	2.6400E-04
Best	1.7351E+00	1.6991E+00	1.7627E+00	1.6688E+00	**1.5170E+00**	1.5739E+00	1.6420E+00
Worst	1.9696E+00	1.7627E+00	1.8264E+00	1.7938E+00	**1.5455E+00**	1.6818E+00	1.6818E+00

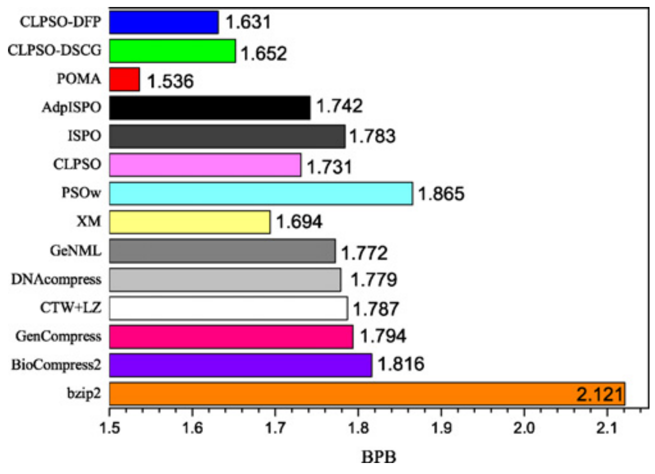


Fig. 11. Mean BPB of the algorithms on the DNA sequence data.

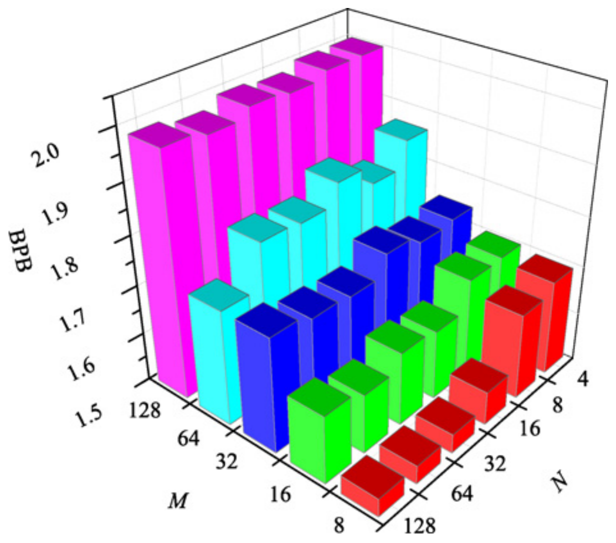


Fig. 12. Effects of code vector size and codebook size to the performance of POMA on the benchmark DNA sequences.

2) *Effect of Discrete Mapping Range:* Next, the impact of the discrete mapping range $[-\hat{X}, \hat{X}]$ on POMA is investigated for $\hat{X} = \{0.1, 1, 5, 10, 20, 20, 50, 100\}$. The mapping ranges and corresponding average fitness values across 30 independent runs are reported in Table VI. An ANOVA test on the performances of over all the configurations suggests their differences are not statistically significant at confidence level of 95%. Hence the results indicate that the search performance of POMA is insensitive to $\hat{X}$. One explanation for this observation is that the moving paces of particles defined in both CLPSO and ISPO are scaled to $\hat{X}$ and the fitness evaluation based on phenotype is robust to the changes in genotype due to the discrete mapping. The range of [−10, 10] used in this paper is defined empirically for it does not affect too much to the final performance.

VI. DISCUSSION AND FUTURE WORK

A. Running Time

Generally, EAs are known to consume more computational effort than other traditional compression algorithms. However, the training time of the codebook is not of a major concern of DNA sequence compression, since the training is often performed offline so that the optimal codebook need only be identified once and used repeatedly. In practice, after the codebook is identified, it is stored and coupled together with the sequences as meta-data to provide efficient compression/decompression to end users. After the one-off time-consuming search for the optimal codebook has been done, compression can be performed in a matter of seconds using the attained codebook. Here, POMA takes 1268 s to optimize the codebook of the 11 DNA sequences on a personal computer with a duo core 2.4 GHz CPU and 2 GB memory. Using the codebook, the compression/decompression of the same 11 DNA sequences can be conducted in merely 3 s. Since all EAs are subjected to the same stopping criteria, i.e., a consistent of maximum fitness function calls, and the fitness evaluation on the DNA sequence compression forms the main cost of the search computation overheads, the algorithms are expected to consume similar CPU time. The running times of PSOw (1250 s), CLPSO (1253 s), ISPO (1432 s), AdpISPO (1228 s), CLPSO-DFP (1343 s), and CLSPO-DSCG (1298 s) to identify the optimal codebook are all observed to fall in the same scale as POMA.

B. Mining Codebook as Meme(s)

One future work of this paper is to mine the codebook as meme(s), which are defined as building blocks of reusable information encoded in computational representation in the context of memetic computing [35]. Since DNA sequences derived from related organisms/species share very high similarity, the optimal codebooks for the related organisms/species are also highly similar or even identical with each other. In this regard, the codebook identified by POMA can thus be represented as meme(s) and reused for compressing unseen sequences from related organisms/species. Particularly, the identified codebook can be transferred directly as reference dictionary or initialization seeds for the search of new codebooks in other homogeneous organisms/species. With the ability of abstracting meme(s) from the existing DNA sequences, the new compression technique can considerably

TABLE VI
MEAN BEST FITNESS VALUES OF 30 INDEPENDENT RUNS FOR DIFFERENT MAPPING RANGES

$-\hat{X}, \hat{X}$	[−0.1, 0.1]	[−1, 1]	[−5, 5]	[−10, 10]	[−20, 20]	[−50, 50]	[−100, 100]	ANOVA(F)	P-Value
Fitness	1.5347E+00	1.5355E+00	1.5327E+00	1.5332E+00	1.5386E+00	1.5344E+00	1.5331E+00	1.622	1.459E-01

speed up the search convergence to the optimal codebook on new sequences. POMA-based DNA sequence compression can also complement the current reference-based genomic compression methods and lead to a candidate general solution for the compression of various high-throughput sequencing data in whole genome scale.

VII. CONCLUSION

In this paper, a novel approach of DNA sequences compression, which takes into consideration all sequences simultaneously in a single entity, has been proposed and described. Four fragment repeat patterns, i.e., direct, mirror, pairing, and inverted repeats were considered and represented with a new ARV model. A novel adaptive POMA was then proposed for the design of ARV codebooks that are effective for maximal DNA sequence compression. Particularly, POMA manifests as a hybridization of the CLPSO as global search with the AdpISPO as local search to reap the benefits of explicit exploration and exploitation. Comparison studies of POMA to other counterpart intelligent search algorithms including PSOw, CLPSO, ISPO, and AdpISPO were carried out on 29 benchmark functions and 11 real DNA sequence data sets. The results obtained suggested that POMA performed better or competitively to the other algorithms. POMA also obtained improved BPB than other representative state-of-the-art DNA sequence compression algorithms on DNA sequence data, and thus serving as a competitive alterative for reducing the storage space requirement of current exponentially increasing DNA sequence data.

REFERENCES

[1] M. L. Metzker, “Sequencing technologies: The next generation,” *Nat. Genet.*, vol. 11, no. 1, pp. 31–46, 2010.

[2] M. Janitz, *Next-Generation Genome Sequencing: Towards Personalized Medicine*. Weinheim, Germany: Wiley-VCH, 2008.

[3] S. Bennett, C. Barnes, A. Cox, L. Davies, and C. Brown, “Toward the $1000 human genome,” *Pharmacogenomics*, vol. 6, no. 4, pp. 373–382, Jun. 2005.

[4] R. Giancarlo, D. Scaturro, and F. Utro, “Textual data compression in computational biology: A synopsis,” *Bioinformatics*, vol. 25, no. 13, pp. 1575–1586, 2009.

[5] J. J. Liang, P. N. Suganthan, and K. Deb, “Novel composition test functions for numerical global optimization,” in *Proc. IEEE SIS*, Jun. 2005, pp. 68–75.

[6] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[7] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in *Proc. IEEE ICNN*, Nov.–Dec. 1995, pp. 1942–1948.

[8] Y. Shi and R. C. Eberhart, “A modified particle swarm optimizer,” in *Proc. IEEE Int. CEC*, May 1998, pp. 69–73.

[9] J. Kennedy, “Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance,” in *Proc. IEEE Int. CEC*, vol. 3. Jul. 1999, pp. 1931–1938.

[10] J. Kennedy and R. Mendes, “Population structure and particle swarm performance,” in *Proc. IEEE Int. CEC*, May 2002, pp. 1671–1676.

[11] R. Mendes, “Population topologies and their influence in particle swarm performance,” Ph.D. dissertation, Departamento de Informatica, Escola de Engenharia, Universidade do Minho, Braga, Portugal, 2004.

[12] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, 1st ed. Chichester, U.K.: Wiley, 2005.

[13] M. Clerc, *Particle Swarm Optimization*, 1st ed. London, U.K.: Wiley-ISTE, 2006.

[14] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization: An overview,” *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.

[15] M. A. M. de Oca, T. Stutzle, M. Birattari, and M. Dorigo, “Frankenstein’s PSO: A composite particle swarm optimization algorithm,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1120–1131, Oct. 2009.

[16] P. Moscato, *Memetic Algorithms: A Short Introduction*. London, U.K.: McGraw-Hill, 1999, pp. 219–234.

[17] T. Matsumoto, K. Sadakane, and H. Imai, “Biological sequence compression algorithms,” in *Proc. GIW*, 2000, pp. 43–52.

[18] S. Grumbach and F. Tahi, “Compression of DNA sequences,” in *Proc. DCC*, 1993, pp. 340–350.

[19] S. Grumbach and F. Tahi, “A new challenge for compression algorithms: Genetic sequences,” *Inform. Process. Manag.*, vol. 30, no. 6, pp. 875–886, 1994.

[20] X. Chen, S. Kwong, and M. Li, “A compression algorithm for DNA sequences and its applications in genome comparison,” in *Proc. 4th Annu. Int. Conf. Computat. Molecular Biol. (RECOMB)*, 2000, pp. 107–117.

[21] X. Chen, M. Li, B. Ma, and T. John, “DNACompress: Fast and effective DNA sequence compression,” *Bioinformatics*, vol. 18, no. 2, pp. 1696–1698, 2002.

[22] G. Korodi and I. Tabus, “DNA sequence compression-based on the normalized maximum likelihood model,” *IEEE Signal Process. Mag.*, vol. 24, no. 1, pp. 47–53, Jan. 2007.

[23] B. Beresford-Smith, T. Conway, S. Kuruppu, and J. Zobel, “Repetition-based compression of large DNA datasets,” in *Proc. 13th Annu. Int. Conf. Res. Computat. Molecular Biol. (RECOMB)*, 2009, poster 16.

[24] D. Loewenstern and P. N. Yianilos, “Significantly lower entropy estimates for natural DNA sequences,” *J. Comput. Biol.*, vol. 6, no. 1, pp. 125–142, 1999.

[25] L. Allison, T. Edgoose, and T. I. Dix, “Compression of strings with approximate repeats,” in *Proc. 6th Int. Conf. ISMB*, 1998, pp. 8–16.

[26] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, “A simple statistical algorithm for biological sequence compression,” in *Proc. DCC*, 2007, pp. 43–52.

[27] S. Kuruppu, S. J. Puglisi, and J. Zobel, “Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval,” in *Proc. 17th Int. Symp. SPIRE*, 2010, pp. 201–206.

[28] M. C. Brandon, D. C. Wallace, and P. Baldi, “Data structures and compression algorithms for genomic sequence data,” *Bioinformatics*, vol. 25, no. 14, pp. 1731–1738, 2009.

[29] S. Christley, Y. Lu, C. Li, and X. Xie, “Human genomes as email attachments,” *Bioinformatics*, vol. 25, no. 2, pp. 274–275, Jan. 2009.

[30] K. Daily, P. Rigor, S. Christley, X. Xie, and P. Baldi, “Data structures and compression algorithms for high-throughput sequencing technologies,” *BMC Bioinform.*, vol. 11, no. 1, p. 514, Oct. 2010.

[31] M. H.-Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney, “Efficient storage of high throughput DNA sequencing data using reference-based compression,” *Genome Res.*, vol. 21, no. 5, pp. 734–740, 2011.

[32] C. Wang and D. Zhang, “A novel compression tool for efficient storage of genome resequencing data,” *Nucleic Acids Res.*, vol. 39, no. 7, p. e45, 2011.

[33] C. Kozanitis, C. Saunders, S. Kruglyak, V. Bafna, and G. Varghese, “Compressing genomic sequence fragments using slimgene,” in *Proc. 14th Annu. Int. Conf. Res. Computat. Molecular Biol. (RECOMB)*, 2010, pp. 310–324.

[34] S. D. Kahn, “On the future of genomic data,” *Science*, vol. 331, no. 6018, pp. 728–729, Feb. 2011.

[35] Y. S. Ong, M. H. Lim, and X. S. Chen, “Research frontier: Memetic computation: Past, present and future,” *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.

[36] B. Liu, L. Wang, and Y. H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 1, pp. 18–27, Feb. 2007.

[37] S. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Comput. J.*, vol. 1, no. 1, pp. 69–83, 2009.

[38] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1151–1166, Oct. 2009.

[39] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 1, pp. 62–69, Feb. 2007.

[40] H. Soh, Y. S. Ong, Q. C. Nguyen, Q. H. Nguyen, M. S. Habibullah, T. Hung, and J.-L. Kuo, "Discovering unique, low-energy pure water isomer: Memetic exploration, optimization and landscape analysis," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 419–437, Jun. 2010.

[41] R. Ruiz-Torrubiano and A. Suárez, "Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 92–107, May 2010.

[42] C. Aranha and H. Iba, "The memetic tree-based genetic algorithm and its application to portfolio optimization," *Memetic Comput. J.*, vol. 1, no. 2, pp. 139–151, 2009.

[43] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 54–65, May 2010.

[44] R. J. Meuth, E. Saad, D. Wunsch, and J. Vian, "Memetic mission management," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 32–40, May 2010.

[45] L. Jiao, M. Gong, S. Wang, and B. Hou, "Natural and remote sensing image segmentation using memetic computing," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 78–91, May 2010.

[46] G. Acampora, M. Gaeta, and V. Loia, "Exploring e-learning knowledge through ontological memetic agents," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 66–77, May 2010.

[47] Z. Zhu, Y. S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 1, pp. 70–76, Feb. 2007.

[48] Z. Zhu, Y. S. Ong, and M. Dash, "Markov blanket embedded genetic algorithm for gene selection," *Patt. Recog.*, vol. 40, no. 11, pp. 3236–3248, 2007.

[49] Z. Zhu, S. Jia, and Z. Ji, "Towards a memetic feature selection paradigm," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 41–53, May 2010.

[50] Z. Zhu, Y. S. Ong, and M. Zurada, "Simultaneous identification of full class relevant and partial class relevant genes," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 7, no. 2, pp. 263–277, Apr.–Jun. 2010.

[51] S. D. Handoko, C. K. Kwoh, and Y. S. Ong, "Feasibility structure modeling: An effective chaperon for constrained memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 740–758, Oct. 2010.

[52] M. N. Le, Y. S. Ong, Y. Jin, and B. Sendhoff, "Lamarckian memetic algorithms: Local optimum and connectivity structure analysis," *Memetic Comput. J.*, vol. 1, no. 3, pp. 175–190, 2009.

[53] Q. C. Nguyen, Y. S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, Jun. 2009.

[54] R. Meuth, M. H. Lim, Y. S. Ong, and D. C. Wunsch, II, "A proposition on memes and meta-memes in computing for higher-order learning," *Memetic Comput. J.*, vol. 1, no. 2, pp. 85–100, 2009.

[55] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern. B*, vol. 36, no. 1, pp. 141–152, Feb. 2006.

[56] T. Yoshida, H. Ishibuchi, and T. Murata, "Balance between genetic search and local search in memetic algorithm for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.

[57] P. Merz, "On the performance of memetic algorithms in combinatorial optimization," in *Proc. GECCO Workshop*, 2001, pp. 168–173.

[58] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, Faculty Comput., Math., Eng., Univ. West England, Bristol, U.K., 2002.

[59] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Comput. J.*, vol. 2, no. 2, pp. 87–110, 2010.

[60] W. Jakob, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Comput. J.*, vol. 2, no. 3, pp. 201–218, 2010.

[61] N. Krasnogor, "Memetic algorithms," in *Handbook of Natural Computation* (Series in Natural Computing). Berlin/Heidelberg, Germany: Springer, 2011.

[62] J. S. Vesterstrøm, J. Riget, and T. Krink, "Division of labor in particle swarm optimization," in *Proc. IEEE CEC*, May 2002, pp. 1570–1575.

[63] R. Poli and C. R. Stephens, "Constrained molecular dynamics as a search and optimization tool," in *Proc. 7th Eur. Conf. Genet. Program. (EuroGP)*, 2004, pp. 150–161.

[64] Y. Petalas, K. Parsopoulos, and M. Vrahatis, "Memetic particle swarm optimization," *Ann. Oper. Res.*, vol. 156, no. 1, pp. 99–127, 2007.

[65] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 1, pp. 42–50, Feb. 2007.

[66] S. Ono and S. Nakayama, "Multi-objective particle swarm optimization for robust optimization and its hybridization with gradient search," in *Proc. IEEE CEC*, May 2009, pp. 1629–1636.

[67] R. Gupta, A. Mittal, and S. Gupta, "An efficient algorithm to detect palindromes in DNA sequences using periodicity transform," *Signal Process.*, vol. 86, no. 8, pp. 2067–2073, 2006.

[68] C. Bi, J. S. Leeder, and C. A. Vyhlidal, "A comparative study on computational two-block motif detection: Algorithms and applications," *Mol. Pharmaceut.*, vol. 5, no. 1, pp. 3–16, 2008.

[69] S. Wu and U. Manber, "Fast text searching: Allowing errors," *Commun. ACM*, vol. 35, no. 10, pp. 83–91, 1992.

[70] Z. Ji, H. Liao, Y. Wang, and Q. H. Wu, "A novel intelligent particle optimizer for global optimization of multimodal functions," in *Proc. IEEE CEC*, Sep. 2007, pp. 3272–3275.

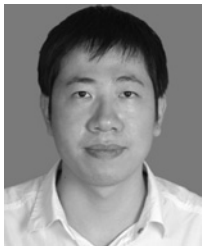
[71] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[72] M. Osborne. (2003). *Predicting DNA Sequences Using a Backoff Language Model* [Online]. Available: http://homepages.inf.ed.ac.uk/miles/dna-backoff.ps.gz

[73] H. P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.

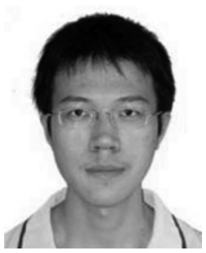
[74] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, vol. 7, no. 4, pp. 303–307, 1964.

[75] Y. S. Ong and A. Keane, "Meta-Lamarckian learning in memetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.



Zexuan Zhu received the B.S. degree in computer science and technology from Fudan University, Shanghai, China, in 2003, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, in 2008.

He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His current research interests include computational intelligence, machine learning, and bioinformatics.



Jiarui Zhou received the Masters degree in pattern recognition and intelligent systems from Shenzhen University, Shenzhen, China, in 2010. He is currently pursuing the Ph.D. degree in biomedical engineering from Zhejiang University, Zhejiang, China.

His current research interests include particle swarm optimization, memetic algorithms, and DNA sequence compression techniques.

Zhen Ji (M'04) received the B.E. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1994 and 1999, respectively.

He is currently a Professor with the Department of Computer Science, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. In 2001, 2003, and 2004, he was an Academic Visitor with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, U.K. Since 2002, he has been the Director of the Texas Instruments DSPs

Laboratory, Shenzhen University. His current research interests include digital image processing, computational intelligence, bioinformatics, and digital signal processors.

Yu-Hui Shi (SM'98) received the Ph.D. degree in electronic engineering from Southeast University, Nanjing, China, in 1992.

He is currently a Professor with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China. He is the Director of the Research and Postgraduate Office, Xi'an Jiaotong-Liverpool University. He is an Adjunct Professor with Indiana University-Purdue University, Indianapolis, Southeast University, and Jiangsu University, Zhenjiang, China. His current research interests include computational intelligence techniques (including swarm intelligence) and their applications.

Dr. Shi is the Editor-in-Chief of the *International Journal of Swarm Intelligence Research* and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He is the Chair of the IEEE Task Force on Swarm Intelligence.