Short Adjacent Repeat Identification Based on Chemical Reaction Optimization

Jin Xu[1], Albert Y.S. Lam[2], Victor O.K. Li[1], Qiwei Li[3], and Xiaodan Fan[3]
[1]Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong
[2]Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720, USA
[3]Department of Statistics, The Chinese University of Hong Kong, Sha Tin, Hong Kong
xujin@eee.hku.hk, ayslam@eecs.berkeley.edu, vli@eee.hku.hk, qwli@sta.cuhk.edu.hk, xfan@sta.cuhk.edu.hk

***Abstract*—The analysis of short tandem repeats (STRs) in DNA sequences has become an attractive method for determining the genetic profile of an individual. Here we focus on a more general and practical issue named short adjacent repeats identification problem (SARIP), which is extended from STR by allowing short gaps between neighboring units. Presently, the best available solution to SARIP is BASARD, which uses Markov chain Monte Carlo algorithms to determine the posterior estimate. However, the computational complexity and the tendency to get stuck in a local mode lower the efficiency of BASARD and impede its wide application. In this paper, we prove that SARIP is NP-hard, and we also solve it with Chemical Reaction Optimization (CRO), a recently developed metaheuristic approach. CRO mimics the interactions of molecules in a chemical reaction and it can explore the solution space efficiently to find the optimal or near optimal solution(s). We test the CRO algorithm with both synthetic and real data, and compare its performance in mode searching with BASARD. Simulation results show that CRO enjoys dozens of times, or even a hundred times shorter computational time compared with BASARD. It is also demonstrated that CRO can obtain the global optima most of the time. Moreover, CRO is more stable in different runs, which is of great importance in practical use. Thus, CRO is by far the best method on SARIP.**

***Index Terms*—Short adjacent repeats, Chemical Reaction Optimization, maximum a posteriori**

I. Introduction

Tandemly repeated DNA sequences are effective genetic markers for mapping studies in many fields, such as disease diagnosis, human identity testing, etc. Specifically, in the last decade biological scientists have been attracted to STRs, which normally have short pattern widths. STRs are enriched around chromosomal centromeres. They are highly polymorphic and especially suitable for personal identification [1]. In STR, an approximate pattern of nucleotides repeated two or more times in a head-tail manner [2], e.g., the 15-base pairs (bp) sequence of "AGGCTAGGCTAGGCT" represents 3 head-tail copies of the pentamer "AGGCT". However, due to sporadic mutation errors in DNA replication, short gaps are often found between neighboring units. This extended form of STR is called short adjacent repeats (SAR), proposed by Li et al. [3]. In the above example, the sequence may turn into "AGGCTgAGGCTAGGCT", where "g" caused by an insertion mutation is the gap between the first and second copies. Since STR is only a special case of SAR where the length of any gap is zero, in our paper, we directly target at the SAR identification problem (SARIP). Note that SARIP does not consider insertions and deletions within a repeat unit.

BASARD (Bayesian Approach for Short Adjacent Repeat Detection) [4] is the best method for SARIP so far. It is based on a probabilistic generative model driven by the motif matrix. To reduce the computational time and improve the efficiency, BASARD employs a collapsing technique [5] in their Markov chain Monte Carlo (MCMC) moves. Yet there are still two weaknesses in BASARD: (1) The computational time remains very long, and this often becomes unacceptable when the size of the input data is large; (2) Since it evolves via a single Markov chain, BASARD is easily stuck at local optima. To tackle these two drawbacks, based on BASARD, three evolutionary Monte Carlo (EMC) schemes were proposed in [6]. Although EMC schemes are relatively good at solving the local optima problem, the reduction in computational time is not significant. In fact, they can only reduce the time by roughly one half, but require much more computer resources.

Chemical Reaction Optimization (CRO) is recently introduced by Lam and Li [7]. It is a multi-purpose evolutionary metaheuristic approach mimicking the interactions of molecules in a chemical reaction. It has found applications in many problems, including the Quadratic Assignment Problem [7], continuous benchmark functions [8], Grid Scheduling Problem [9], the population transition problem in peer-to-peer live streaming [10], and artificial neural network training [11], etc. The state-of-the-art developments of CRO can be found in [12]. In this paper, we propose a CRO-based algorithm to solve SARIP. Different from BASARD that only manipulates one single Markov chain, CRO is a population-based algorithm, manipulating multiple molecules (solutions) simultaneously, thus enhancing its ability to escape from local optima. Besides, although we also employ the same collapsing technique as in BASARD, here we only target the maximum a posteriori (MAP) instead of the whole posterior distribution. In other words, we substitute CRO for Gibbs sampling and Metropolis-Hasting in BASARD, and this can save a lot of computational time as shown in our simulation results.

The rest of the paper is organized as follows. In Section II, we describe the problem of SARIP and present its objective

978-1-4673-1509-8/12/$31.00 ©2012 IEEE

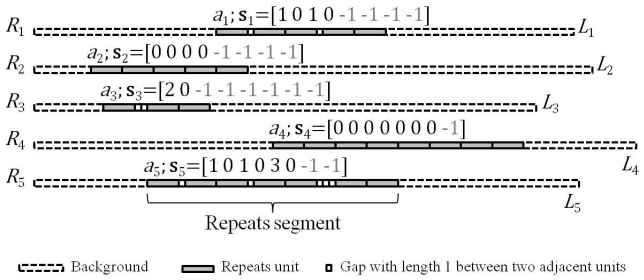

Fig. 1. Schematic diagram of the model under the setting $N = 5$ and $\Omega = 9$

function. In Section III, we prove that SARIP is an NP-hard problem. Then in Section IV, we propose a CRO-based algorithm for solving SARIP. In Section V, we discuss the experimental results on both synthetic and real data. Finally, conclusions and potential future work are given in Section VI.

II. Problem Descriptions

SARIP is among the most difficult combinatorial optimization problems in bioinformatics. It is easy to state but hard to solve because of the extremely large size of DNA sequences. Moreover, SARIP has important applications in other fields, such as data mining and analysis in detecting repetitive patterns in speeches, texts, or images.

Consider a set of N DNA sequences, the lengths of which are denoted by $\{L_1, L_2, \cdots, L_i, \cdots, L_N\}$. Thus, the whole input data can be expressed as: $\mathbf{R} = \{R_1; R_2; \cdots; R_i; \cdots; R_N;\}$, where $R_i = \{r_{i,1}, r_{i,2}, \cdots, r_{i,L_i}\}$, and each element $r_{i,j}$ in $\mathbf{R}$ is a letter chosen from $\{A, T, C, G\}$, which represent the four basic nucleotides in DNA. The region in the sequence where the same repeat units cluster is called a repeat segment, and we assume that each sequence contains at least one repeat segment. In SARIP, we aim to find the most probable repeat segment location and its corresponding structure for each sequence. They are defined as the repeat segment starting position $\mathbf{A} = \{a_1, a_2, \cdots, a_i, \cdots, a_N\}$ and the structure $\mathbf{S} = \{S_1; S_2; \cdots; S_i; \cdots; S_N\}$, respectively. For parameter $\mathbf{A}$, each a_i is an integer selected from $[1, L_i]$, while for parameter $\mathbf{S}$, each S_i is a vector and can be written as $[g_{i,1}, g_{i,2}, \cdots, g_{i,\Omega_i-1}, -1, \cdots, -1]$, where Ω_i is the copy number in sequence R_i, and the variable $g_{i,\omega}$ is the gap length between the ω-th and the $(\omega+1)$-th repeat units. For calculation convenience, we expand each S_i to the same dimension by filling blanks with the trivial value -1 from the Ω_i-th to the $(\Omega-1)$-th element, where Ω is the maximum allowed copy number in all sequences. Therefore, the tuple $\{\mathbf{A}, \mathbf{S}\}$ represents a solution of our problem.

For example, an instance of the problem with five sequences is shown in Fig. 1 (adapted from [4]). The maximum allowed copy number Ω equals 9. In each sequence, a repeat segment consists of multiple repeat units (gray blocks), which are inserted with gaps (white blocks with solid borderline), while the background area is represented by the dotted borderline. Moreover, the repeat segment starting position a_i and its structure S_i is shown above each sequence.

In our algorithm, we aim to seek the unnormalized MAP, which is a mode of the posteriori distribution. Usually, MAP is employed as a point estimate of an unobserved quantity based on empirical data. In short, the bigger the value of the posteriori probability $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$, the more precisely we find the repeat units. Thus, we search for a tuple $[\mathbf{A}, \mathbf{S}]$ which can maximize $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$. According to [4], the objective function of SARIP can be mathematically written as follows:

$$maximize \quad P(\mathbf{A}, \mathbf{S}|\mathbf{R}) = \zeta_1^{\sum_{i=1}^{N}\Omega_i}\zeta_2^{\sum_{i=1}^{N}\sum_{\omega=1}^{\Omega_i-1} g_{i,\omega}} \times \frac{\Gamma(|\alpha|)\Gamma(\mathbf{h}(\mathbf{R}_{U^c})+\alpha)}{\Gamma(\alpha)\Gamma(|\mathbf{h}(\mathbf{R}_{U^c})|+|\alpha|)} \prod_{j=1}^{J} \frac{\Gamma(|\beta_j|)\Gamma(\mathbf{h}(\mathbf{R}_{U(j)})+\beta_j)}{\Gamma(\beta_j)\Gamma(|\mathbf{h}(\mathbf{R}_{U(j)})|+|\beta_j|)},$$

where both of the constants ζ_1 and ζ_2 are in the range of $(0, 1]$, and J is the pattern width. α and β_j are the parameters for prior distribution and their elements are all configured as 1. Given an input data $\mathbf{R}$ and a trial tuple $[\mathbf{A}, \mathbf{S}]$, we use the set $\mathbf{R}_U$ to denote the nucleotides in all repeat units. In particular, $\mathbf{R}_{U(j)}$ could be used to represent the set of nucleotides exactly in the jth ($j = 1, 2, \cdots, J$) position in all repeat units, whilst $\mathbf{R}_{U^c}$ lists all the nucleotides in background regions. $\mathbf{h}(\bullet)$ is the counting function [5]. Note that in our problem both $\mathbf{h}(\mathbf{R}_{U^c})$ and $\mathbf{h}(\mathbf{R}_{U(j)})$ are 1×4 vectors, and they record the number of each type of nucleotide in $\mathbf{R}_{U^c}$ and $\mathbf{R}_{U(j)}$, respectively. Additionally, the absolute value of a vector is the sum of all the elements within the vector. If the variable is a non-negative integer, the above gamma function Γ can be transformed to a factorial function, and we also note that $P(\mathbf{A}, \mathbf{B}|\mathbf{R})$ will be an extremely small value. To facilitate processing by the computer, we calculate the natural logarithm of $P(\mathbf{A}, \mathbf{B}|\mathbf{R})$ instead. Meanwhile, since the energy in CRO should be positive, we reformulate the objective function as:

$$minimize \quad F = -log\ P(\mathbf{A}, \mathbf{S}|\mathbf{R}).$$

III. NP-hardness of SARIP

The problem of finding the MAP in SARIP appears to be difficult to solve. The number of possible combinations of $[\mathbf{A}, \mathbf{S}]$ is approximated by $[(G+1)^{\Omega}L]^N$, where G is the maximum allowed gaps and we assume that all the sequences have the same length L. For example, when we assign $2, 10, 1000$, and 10 to the arguments G, Ω, L, and N, respectively, the size of the solution space is roughly equal to 5.15×10^{77}. In fact, in the following we will show that SARIP belongs to a class of problems called non-deterministic polynomial time hard (NP-hard) problems.

Before we proceed to the proof, let us briefly review some concepts from computational complexity theory [13]. This theory is only applied to the decision problems, which take either true or false as an answer. In general, all the optimization problems can be easily stated as a corresponding decision problem [13]. A set of decision problems solvable in polynomial time by deterministic algorithms belong to the P class. On the other hand, all the problems that can be checked

in polynomial time but solved by non-deterministic algorithms are called NP. Obviously, we have $P \subseteq NP$. Moreover, the hardest problems in NP is named NP-complete problems, which means we cannot obtain the optimal solution(s) in a polynomial time unless $P = NP$. Informally, a problem is called NP-hard if it is at least as hard as the NP-complete problems. Hence, it can be easily found that NP-complete problems $\subseteq$ NP-hard problems. In this paper, we will show that SARIP is an NP-hard problem via proving that it belongs to the NP-complete class.

The proof of the NP-completeness for a given problem can be accomplished by a polynomial reduction from a known NP-complete problem [13]. Here we reduce SARIP from the bounded knapsack problem, which is a well-known NP-complete problem [13]. The bounded knapsack problem can be stated as follows:

INSTANCE: A finite set of items, with nonnegative integer values $\{v_1, v_2, \cdots, v_n\}$ and weights $\{w_1, w_2, \cdots, w_n\}$, positive integers V and W, and a set of the maximum quantities $\{c_1, c_2, \cdots, c_n\}$.

QUESTION: Is there an assignment of x_i, $i = 1, 2, \cdots, n$ and each $x_i \in \{0, 1, \cdots, c_i\}$ such that $\sum_{i=1}^{n} w_i x_i \leq W$ and $\sum_{i=1}^{n} v_i x_i \geq V$?

Note that the above knapsack problem remains NP-complete if $v_i = w_i, i = 1, 2, \cdots, n$ and $V = W$ [13]. In other words, given a set of nonnegative integer numbers $\{a_1, a_2, \cdots, a_n\}$ and a positive integer B, the problem of finding a solution set $\{x_1, x_2, \cdots, x_n\}$, each $x_i \in \{0, 1, \cdots, c_i\}$ such that $\sum_{i=1}^{n} a_i x_i = B$ is NP-complete.

Lemma: The problem defined by the following pair, called the bounded subset sum problem, is an NP-complete problem.

INSTANCE: Finite set $\{u_1, u_2, \cdots, u_n\}$, each $u_i \in \Delta_i$, where Δ_i is a set of possible integer values for u_i, and an integer K.

QUESTION: Is there a choice of u_i, $i = 1, 2, \cdots, n$, such that $\sum_{i=1}^{n} u_i = K$?

Proof: Consider a Turing machine M that on any given K of the problem, nondeterministically assign values from Δ_i to each u_i, $i = 1, 2, \cdots, n$, and check whether $\sum_{i=1}^{n} u_i = K$ holds. Clearly, the check process can be done in polynomial time. Therefore, the bounded subset sum problem belongs to NP class.

Without loss of generality, consider an instance Q of bounded knapsack problem, and let the corresponding values and weights be $v_i = w_i = a_i$, $i = 1, 2, \cdots, n$. Suppose $\{x_1, x_2, \cdots, x_n\}$ denote the variables and $x_i \in \{0, 1, \cdots, c_i\}$, $i = 1, 2, \cdots, n$. Thus, the equation for Q can be represented as follows:

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = K$$

Similarly, the equation for the Q can also be written in the following form:

$$u_1 + u_2 + \cdots + u_n = K,$$

where $u_i = a_i x_i$ and $\Delta_i = \{0, a_i, 2a_i, \cdots, c_i a_i\}$, $i = 1, 2, \cdots, n$. As a result, the instance Q of the bounded knapsack problem is satisfiable if and only if the above instance $\{u_1, u_2, \cdots, u_n, K\}$ has a positive solution. Obviously, this transducer can be constructed in polynomial time. Hence, the bounded subset sum problem is NP-complete. ■

Theorem: SARIP is NP-complete.

Proof: The core part of the objective function for SARIP is to count the number of $\{A, T, C, G\}$ in background and repeat units, respectively. More precisely, for each possible solution $\{\mathbf{A}, \mathbf{S}\}$, we will use $[b_A, b_T, b_C, b_G]$ to denote the number of $\{A, T, C, G\}$ in the background region. Meanwhile, $[m_{Ai}, m_{Ti}, m_{Ci}, m_{Gi}]$ is employed to indicate the number of $\{A, T, C, G\}$ at the ith position of repeat units and $i = 1, 2, \cdots, J$, where J is the pattern width of the repeat unit. Note that the calculation for the counting function can be of polynomial time complexity. Hence, given by a tuple $\{\mathbf{A}, \mathbf{S}\}$ and a number K, we can check if the MAP equals K in a polynomial time, which implies that SARIP is in NP class.

Assume that the input has N sequences. For each sequence z, $z = 1, 2, \cdots, N$, we have the following vector for each corresponding $\{\mathbf{A}, \mathbf{S}\}$:

$$V_z = [m_{Az1}, m_{Tz1}, m_{Cz1}, m_{Gz1}, m_{Az2}, m_{Tz2}, m_{Cz2}, m_{Gz2}, \cdots, m_{AzJ}, m_{TzJ}, m_{CzJ}, m_{GzJ}, b_{Az}, b_{Tz}, b_{Cz}, b_{Gz}, g_z],$$

where g_z is the total number of gaps in sequence z. Meanwhile, we denote by Δ_z the set of all possible values for V_z, which are calculated in light of all possible $\{\mathbf{A}, \mathbf{S}\}$ for z. Then SARIP can be transformed to a problem of finding a vector $V (= V_1 + V_2 + \cdots + V_N)$, which maximizes the MAP. Consequently, the proof of NP-completeness for SARIP is equivalent to verify the problem defined by the following two pairs is NP-complete.

INSTANCE: Finite set $\{V_1, V_2, \cdots, V_N\}$, $V_z \in \Delta_z$, where Δ_z is defined as mentioned above, and vector V.

QUESTION: Is there a choice of V_z such that $\sum_{z=1}^{N} V_z = V$?

By substituting the V_z and V with u_i and K in lemma respectively, the problem with the above two pairs is the same as that in lemma. Thus, it suffices to show that SARIP is NP-complete and thereby belongs to NP-hard class. ■

Corollary: The problem of identifying short tandem repeats in multiple sequences is NP-hard. This is a special case of SARIP. Now the solution only has the starting position **A**, and we also remove the g_z from the vector V_z.

By now, we have shown SARIP is NP-hard, which implies that it is a complex combinatorial optimization problem. On the other hand, in the literature, metaheuristic algorithms are usually designed to tackle complex optimization problems where other optimization methods have failed to be either effective or efficient. In many cases, metaheuristic methods can obtain satisfied solutions in a tolerable period of time. There are various metaheuristics, e.g. Simulated Annealing (SA) [14], Genetic Algorithm (GA) [15], Particle Swarm Optimization (PSO) [16], and CRO [7] etc. In [9], authors observed that CRO not only enjoys both the advantages of GA and SA but it is also more flexible. Thus, in this paper,

we will only focus on CRO and apply it to solve SARIP.

IV. The Proposed Algorithm

A. Chemical Reaction Optimization

In this subsection, we give a brief introduction to CRO. For a more detailed description of this algorithm, the readers may refer to [7].

CRO is a population-based metaheuristic used to generate good approximate solutions in global optimization problems. By analogy with the chemical reaction process in which molecules' energy changes step by step and finally reaches the lowest free energy, CRO optimizes a function via iterative trials to improve a set of candidate solutions. Predefined elementary reactions are the crucial components in CRO, and they guide the algorithm to reach the global optimum.

In CRO, each solution can be considered as a molecule, which has certain attributes, including potential energy (*PE*), kinetic energy (*KE*), the molecular structure, the number of hits, etc. Specifically, PE is regarded as the fitness value of the solution, while KE determines the tolerance of accepting a worse solution. In addition, the minimum structure is the best-so-far solution recorded and the number of hits is used in the reaction of decomposition, which will be illustrated later. We assume that all the reactions happen in a closed container, and there is a central energy buffer to store or release energy to molecules. By the conservation of energy, the total amount of energy of the container always keeps constant. Moreover, four types of elementary reactions are employed in the process of CRO and their characteristics are described as follows:

- On-wall ineffective collision: it only involves one molecule and the reaction is triggered when the molecule hits on the wall of the container.
- Decomposition: it also happens when a molecule hits against the wall, but the molecule is decomposed into two molecules.
- Inter-molecular ineffective collision: two molecules interact with each other and generate two new molecules.
- Synthesis: two molecules take part in the reaction and they are combined into a new one.

These four elementary reactions are responsible for generating new candidate solutions from the original ones. The operating mechanisms adopted in these reactions can be varied in different problems. However, there is a general rule for the operator design: from the perspective of the solution space, in the two ineffective collisions, new molecule(s) are generated in the neighborhood of the original one(s), while in decomposition and synthesis reactions, molecule(s) are generated to explore regions far from the original ones. The former corresponds to "intensification" in the search process whereas the latter corresponds to "diversification". Moreover, one advantage of CRO is its flexible structure, which means that in practical uses we can choose some or even only one of the four elementary reactions to implement CRO.

Generally, the CRO algorithm can be divided into three stages, i.e., initialization, iterations, and output. In initialization, a number of solutions are generated according to some rules, and usually randomly. Then the algorithm enters into the second stage, and for each iteration, one of the four elementary reactions is selected based on some criteria. Meanwhile the randomly chosen molecule(s) would try to participate the appointed reaction, and the new molecule(s) would take the place of the original one(s) when the reaction happens. Finally, we output the best solution and its fitness value. The schematic procedure of CRO is presented in Table I.

TABLE I
The Schematic Procedure of the CRO algorithm

Step 1:	Initialize a population of molecules, compute their fitness values as the *PE*s, and set each molecule's other attributes;
Step 2:	Repeat: 2.1: Decide whether it is a unimolecular or an inter-molecular collision, and accordingly select one or two molecules from the population; 2.2: According to the decomposition criterion α or synthesis criterion β, pick an elementary reaction; 2.3: Generate the new molecule(s) according to the relevant operations; 2.4: Replace the original molecule(s) with the new one(s) when the reaction is triggered, and compute the *KE*(s). If the energy is not enough to support the change, ignore the new molecule(s) and keep the original one(s); Until the stopping criterion is met;
Step 3:	Output the best solution;

B. Implementation of CRO on SARIP

In order to apply CRO to SARIP, the most important consideration is the operator design for the four elementary reactions. CRO can obtain good solutions by utilizing well-designed operators, so as to achieve a good balance between intensification and diversification. We list the operators in solving SARIP as follows:

- On-wall ineffective collision: firstly we randomly choose one sequence $i \in [1, N]$, and then renew its starting position a_i and structure S_i consecutively. For updating a_i, a random integer in $[1, L_i - \Omega_i J - \sum_{\omega=1}^{\Omega_i} g_{i,\omega} + 1]$ is selected to be the new a_i', while for S_i, in each iteration we randomly adopt one of the five moves used in BASARD [4], i.e., rear deletion, real insertion, partial shift, front deletion and front insertion, as shown in Fig. 2 (adapted from [4]). Finally, since in solving SARIP the algorithm encounters the phase problem [17], the technique of phase shift [4] is also employed and activated at a certain frequency. Let ω and ω' be the molecules before and after the collision. Then, this elementary reaction can be concisely represented as:
  $\omega : [a_1, a_2, \cdots, \underline{a_i}, \cdots, a_N, S_1, S_2, \cdots, \underline{S_i}, \cdots, S_N] \rightarrow$
  $\omega' : [a_1, a_2, \cdots, \underline{a_i'}, \cdots, a_N, S_1, S_2, \cdots, \underline{S_i'}, \cdots, S_N]$
- Decomposition: a chosen molecule together with a randomly generated molecule constitute the new molecules. This reaction can be shown as:
  $\omega : [a_1, a_2, \cdots, a_N, S_1, S_2, \cdots, S_N] \rightarrow$
  $\omega' : [a_1, a_2, \cdots, a_N, S_1, S_2, \cdots, S_N] +$
  $\omega'' : [a_1', a_2', \cdots, a_N', S_1', S_2', \cdots, S_N']$
- Inter-molecular ineffective collision: a crossover tech-

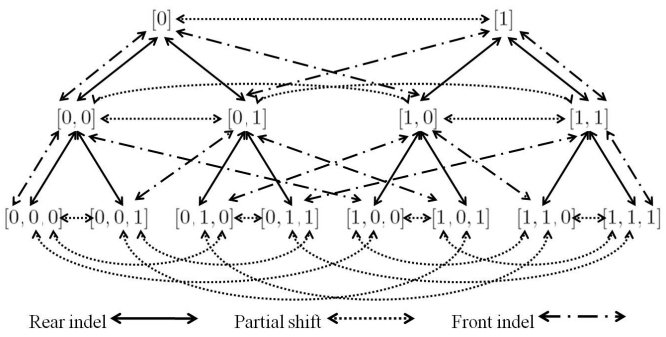

Fig. 2. The full state transition diagram under the setting $G = 1$ and $\Omega = 4$.

nique is employed, which means that the two new molecules are generated by swapping half of the starting positions and structures between the two original molecules. Without loss of generality, when N is an even number, this process can be illustrated as:

$\omega_1 : [a_{11}, a_{12}, \cdots, a_{1N}, S_{11}, S_{12}, \cdots, S_{1N}]+$
$\omega_2 : [a_{21}, a_{22}, \cdots, a_{2N}, S_{21}, S_{22}, \cdots, S_{2N}] \rightarrow$
$\omega_1' : [a_{11}, a_{22}, \cdots, a_{2N}, S_{11}, S_{22}, \cdots, S_{2N}]+$
$\omega_2' : [a_{21}, a_{12}, \cdots, a_{1N}, S_{21}, S_{12}, \cdots, S_{1N}]$

- Synthesis: similar to the operation in inter-molecular ineffective collision but only one new molecule is kept. Concisely, this reaction is:

$\omega_1 : [a_{11}, a_{12}, \cdots, a_{1N}, S_{11}, S_{12}, \cdots, S_{1N}]+$
$\omega_2 : [a_{21}, a_{22}, \cdots, a_{2N}, S_{21}, S_{22}, \cdots, S_{2N}] \rightarrow$
$\omega' : [a_{11}, a_{22}, \cdots, a_{2N}, S_{11}, S_{22}, \cdots, S_{2N}]$

The reactions only happen when the energy requirement is met. For example, suppose ω and ω' are the existing and possible new molecules, respectively. The on-wall ineffective collision can only be triggered when $PE_\omega + KE_\omega \geq PE_{\omega'}$. In other words, the new molecule (solution) ω' will substitute ω in the population if the above energy requirement is satisfied. In the meantime, the kinetic energy for ω' would be calculated by $KE_{\omega'} = (PE_\omega + KE_\omega - PE_{\omega'}) \times \delta$, where $\delta \in [\textit{KELossRate}, 1]$, and *KELossRate* is a parameter predefined to control the *KE* loss. Readers can refer to [7] for the energy requirement and *KE* calculation of other reactions.

Actually, for those problem instances with strong repeat units signal or flat solution space, we only need the on-wall ineffective collision, which is efficient enough to find a good solution. However, when the SAR signal is not so strong or the solution space is bumpy, molecules tend to be stuck at local optima if only on-wall ineffective collision is adopted. In this case, the other three reactions can be used to release the trapped molecule(s). Actually, most of the time, we do not know the signal strength of the input data in advance. Thus, we generally apply CRO with all four reactions to SARIP.

V. Experimental Results

In this section, we firstly describe the simulation environment, including the testing data and the CRO parameters. To have a comprehensive comparison, CRO and BASARD are tested on both synthetic and real data. Then we discuss the experimental results. Finally, we summarize by analyzing the reason behind the different performance of BASRD and CRO.

TABLE II
All three data settings

Single segment synthetic case	$N = 6, L_i = 2000, i = 1, 2, \cdots, N,$ $G = 2, J = 6, \Omega = 15$
Multiple segment synthetic case	$N = 5, L_i = 2000, i = 1, 2, \cdots, N,$ $G = 2, J = 9, \Omega = 9$
Real data case	$N = 24, G = 6, J = 18, \Omega = 20$

TABLE III
Parameter settings for CRO

Parameter	Assigned value
Initial population size	1
α	$100 \times N \times J$
β	$0.01\times$initial minimal fitness
KE loss rate	0.95
MoleColl	0.1
Initial *KE*	initial minimal fitness
Initial central buffer	initial minimal fitness

A. Simulation environment

In nature, DNA sequences are composed of several thousand to over a hundred thousand nucleotide bases, rendering the input data quite large and diverse. Thus, considering that the solution space can be diversified with different input data, we choose two typical cases from the randomly generated synthetic data. One is called the single segment case, where there is only one repeat segment in each sequence, while the other allows multiple repeat segments within a sequence. In our problem, we only try to find the most probable repeat segment in each sequence. So obviously the objective function F has many local optima in the multiple segment case. In order to further demonstrate the superiority of CRO over BASARD, we also test it on a real data case from [4], which includes 24 publicly available DNA sequences of *DRD4* exon III from GenBank [18]. *DRD4* is an important gene and it has some relationship with human cognitive function [19]. Also note that in [4] BASARD has already shown its good performance in solving this real data case when compared with existing methods. All these three data settings are shown in Table II. As an iteration of BASARD contains the exhaustive Gibbs sampling, in the simulation we use 3,000 iterations as the stopping criterion for BASARD [4] and 150,000 iterations for CRO.

The parameter values used in CRO are listed in Table III, and they are set up empirically. According to the parameter analysis in [9], in certain ranges these parameters are not sensitive to the performance of CRO. The population size can automatically change in the process due to the invoked reactions of decomposition and synthesis. For the fair comparison with BASARD, we start CRO with only one single molecule. When this molecule is trapped for a certain number of iterations, decomposition occurs, which results in more than one molecule. Moreover, both CRO and BASARD are coded

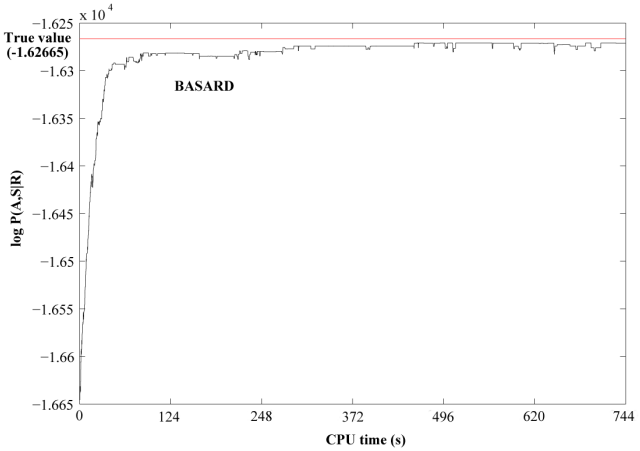

Fig. 3. The trace plot of log $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$ for BASARD in the single segment case.

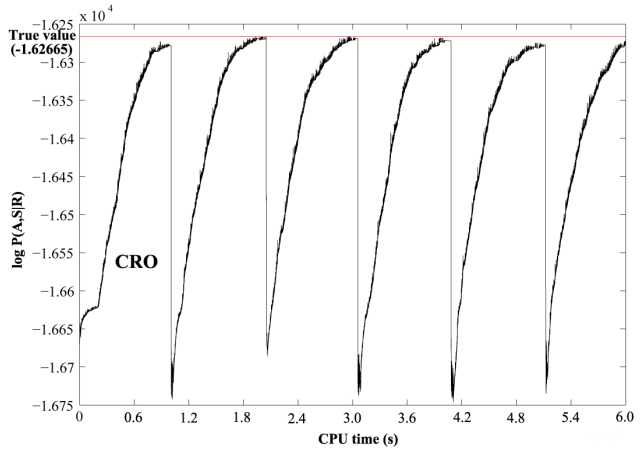

Fig. 4. The trace plot of log $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$ for CRO in the single segment case.

TABLE IV
AVERAGE AND STANDARD DEVIATION VALUES FOR EACH ALGORITHM ON THE SINGLE SEGMENT CASE

	BASARD	CRO
Unnormalized MAP (Avg)	-16271.7	-16267.2
Unnormalized MAP (SD)	6.19	1.98
CPU time(s) (Avg)	745.09	6.27
CPU time (SD)	2.50	0.81

in C++, and they are conducted on the same computer with an Intel Core Duo 2.66 Hz and 2 GB RAM.

B. Results and discussion

CRO and BASARD are both stochastic algorithms, and the results may be different in different runs with the same parameters. Thus, for both CRO and BASARD, we repeat the run 30 times for each of the cases below.

1) Single segment case: Figs. 3 and 4 display one of the 30 simulation runs on the single segment case and they illustrate the trace plots of BASARD and CRO. Note that there may

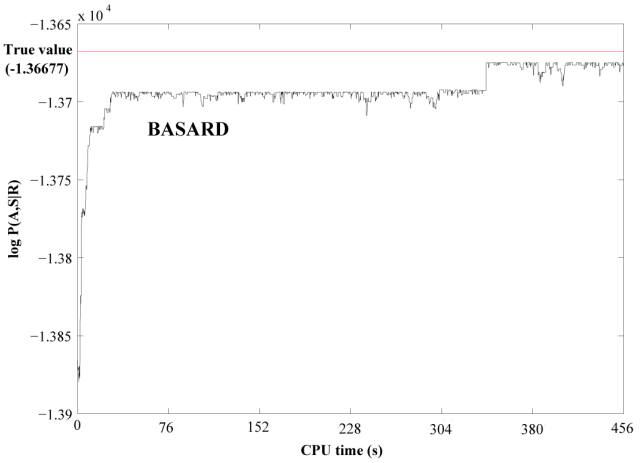

Fig. 5. The trace plot of log $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$ for BASARD in the multiple segment case.

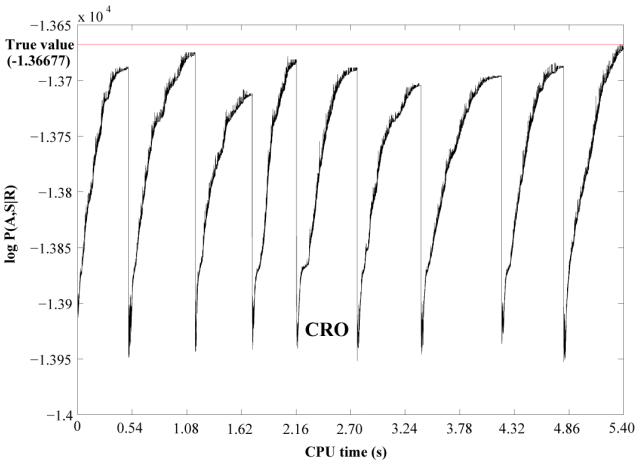

Fig. 6. The trace plot of log $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$ for CRO in the multiple segment case.

be more than one molecule during the CRO evolution, and we only plot the initial molecule's trace in Fig. 4. When a molecule can not find a better solution in its neighborhood for a certain number of iterations, decomposition is triggered. The newly generated molecules have more energy obtained from the central buffer which helps them jump out of a local optimum. In fact, each dramatic fluctuation in Fig. 4 indicates that the molecule is experiencing decomposition or synthesis. Since this is synthetic data, we can calculate the optimal objective function value in advance. In this case, the optimal value is -16266.5. From these two figures, we can observe that BASARD fails to obtain the optimal value, while CRO successfully finds the global optimum.

We also list the average and standard deviation values for the 30 simulation runs in Table IV. On the average, CRO can not only obtain better solutions than BASARD but also enjoys more than 100 times savings in computational time. Moreover, all the standard deviations of CRO are much smaller than those

TABLE V
AVERAGE AND STANDARD DEVIATION VALUES FOR EACH ALGORITHM ON THE MULTIPLE SEGMENT CASE

	BASARD	CRO
Unnormalized MAP (Avg)	-13682.5	-13669.9
Unnormalized MAP (SD)	9.81	3.68
CPU time(s) (Avg)	456.68	5.28
CPU time (SD)	17.99	0.13

of BASARD, which indicates CRO is more stable and reliable.

2) Multiple segment case: This is also a synthetic case but with much more local optima. Fig. 5 and 6 show the convergence traces of BASARD and CRO on this multiple segment case. It can be observed in Fig. 5 that BASARD gets trapped in a local optima for a long time before moving to a region with better solutions. However, BASARD still does not reach the global optimum in 3,000 iterations. The CRO curve also become much more oscillated as shown in Fig. 6, and this is caused by the rugged solution space. Here the optimal value is -13667.7. Similar to the results of the single segment case, CRO is superior to BASARD in both the quality of the solution and the computational time. As shown in Table V, BASARD easily gets stuck in local optima and CRO can achieve the global optimum. Moreover, CRO is 86 times faster than BASARD in terms of computational time.

3) Real data case: In this case, 24 sequences are involved and the repeat pattern width is 18. However, the lengths of these sequences are relatively short ranging from 256 to 512 bps. This indicates that the repeat signal is much stronger than the previous two synthetic cases. Intuitively, the solution space here is large but very flat. Thus, it takes much more time for BASARD and CRO to converge but their curves now are smooth, as show in Fig. 7 and 8. Since it is a real data case and there is no true value known in advance, we show the best value (=-7689.1) found. By comparing the average and standard deviation values of the two algorithms in Table VI, the solutions obtained from CRO are much better than those of BASARD, and CRO shortens the computational time as much as 25 times. In particular, we note that the standard deviation of BASARD are quite large in this case, which means we have to run many times to get a good solution with BASARD in real implementation.

C. Summary

These three tested cases are representative in terms of solution space. Basically, CRO performs much better than BASARD especially in terms of computational time. Note that BASARD is also a heuristic method, and the Gibbs sampling and Metropolis-Hasting schemes guarantee its ergodic search. This means theoretically BASARD will finally reach the global optimum if given sufficient time. Nevertheless, the probability of jumping out a local optimum is quite small in BASARD, which implies it needs a very long computational time to find a good solution. Moreover, the Gibbs sampling in BASARD is also time-consuming as it needs to calculate each probability

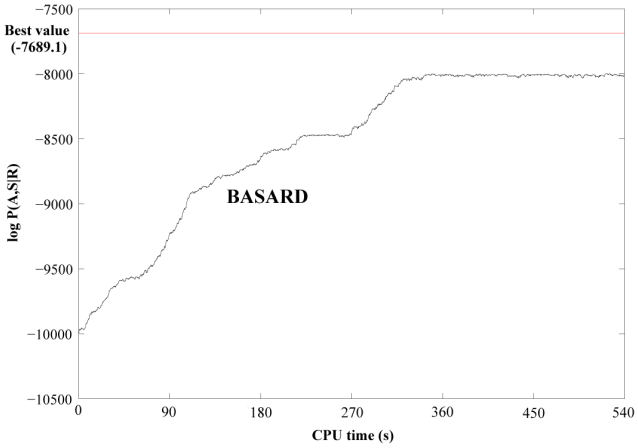

Fig. 7. The trace plot of log $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$ for BASARD in the real data case.

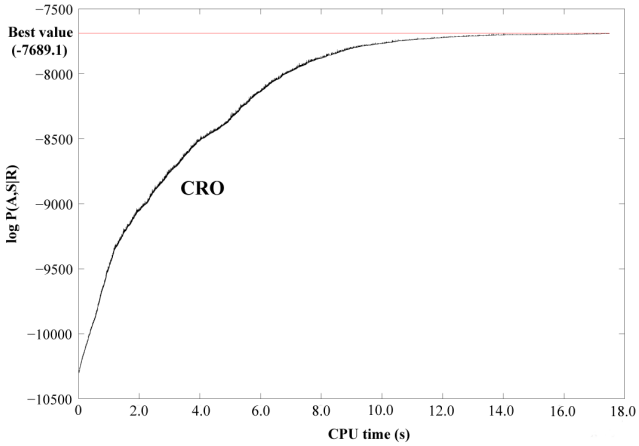

Fig. 8. The trace plot of log $P(\mathbf{A}, \mathbf{S}|\mathbf{R})$ for CRO in the real data case.

TABLE VI
AVERAGE AND STANDARD DEVIATION VALUES FOR EACH ALGORITHM ON THE REAL DATA CASE

	BASARD	CRO
Unnormalized MAP (Avg)	-8024.4	-7701.1
Unnormalized MAP (SD)	693.72	26.22
CPU time(s) (Avg)	464.87	18.48
CPU time (SD)	141.97	0.33

with all starting positions every time. As for CRO, since we adopt randomly generated molecules in decomposition, it is also an ergodic search. Meanwhile, in CRO we use energy conservation to guide the evolution of the molecules, while BASARD is driven by probability. Besides the quality of the solution, the computational time is also a major consideration in real applications. Thus, CRO is a better tool in solving SARIP in terms of mode searching. As BASARD was the best algorithm for the problem before this work, CRO replaces BASARD to be the current best.

VI. Conclusion

Disparity in the number of SAR can explain the polymorphism in a population or relationship between different species, and SARIP has been a hot research topic. In this paper, we apply the CRO algorithm to detect the short adjacent repeats shared by multiple sequences. Our contributions are mainly in three parts: (1) We prove that SARIP is an NP-hard problem, which confirms that metaheuristic approaches may be applicable. (2) We choose the MAP as the objective function and design a suitable CRO-based algorithm which avoids the calculation of the motif matrix and the computationally demanding Gibbs sampling in BASARD; (3) We assess CRO on three different kinds of cases (i.e. single segment case, multiple segment case, and real data case). Compared with the previous best algorithm BASARD, we demonstrate that CRO performs much better in solution quality, computational time, and stability. Hence, we conclude that CRO is so far the best choice for solving SARIP.

Potential future study includes the following: (1) In this paper, we only employ a generic CRO, and we may design more CRO operations to be adaptive to various scenarios of SARIP; (2) We can consider developing parallel CRO for SARIP, which may further shorten the computation time. (3) Locating the multiple repeat segments and accounting for the intra-unit insertions or deletions are also very interesting topics.

Acknowledgment

J. Xu and V. O. K. Li are supported in part by the University of Hong Kong Strategic Research Theme of Information Technology. A. Y. S. Lam is also supported in part by the Croucher Foundation Research Fellowship. Q. Li and X. Fan are partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project no. CUHK400709).

References

[1] A. Edwards, A. Civitello, H. A. Hammond, and C. T. Caskey, "DNA typing and genetic mapping with trimeric and tetrameric tandem repeats," *The American Journal of Human Genetics*, vol. 49, no. 4, pp. 746–756, 1991.

[2] G. Benson, "Tandem repeats finder: A program to analyze DNA sequences," *Nucleic Acids Research*, vol. 27, no. 2, pp. 573–580, 1999.

[3] Q. Li, T. Liang, S. Y. R. Li, and X. Fan, "Bayesian approach for identifying short adjacent repeats in multiple dna sequences," in *Proceedings of the 2010 International Conference on Bioinformatics & Computational Biology (BIOCOMP'10)*, Las Vegas, Nevada, USA, July 2010.

[4] Q. Li, X. Fan, T. Liang, and S. Y. R. Li, "An MCMC algorithm for detecting short adjacent repeats shared by multiple sequences," *Bioinformatics*, vol. 24, no. 13, pp. 1772–1779, May 2011.

[5] J. S. Liu, A. F. Neuwald, and C. E. Lawrence, "Bayesian models for multiple local sequence alignment and Gibbs sampling strategies," *Journal of the American Statistical Association*, 1995.

[6] J. Xu, Q. Li, V. O. K. Li, S. Y. R. Li, and X. Fan, "Improved short adjacent repeat identification using three evolutionary Monte Carlo schemes," *International Journal of Data Mining and Bioinformatics*, accepted for publication.

[7] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 381–399, Jun. 2010.

[8] A. Y. S. Lam, V. O. K. Li, and J. J. Q. Yu, "Real-coded chemical reaction optimization," *IEEE Trans. Evol. Comput.*, accepted for publication.

[9] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1624–1631, Jul. 2010.

[10] A. Y. S. Lam, J. Xu, and V. O. K. Li, "Chemical reaction optimization for population transition in peer-to-peer live streaming," in *Proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2010)*, Barcelona, Spain, Jul. 2010.

[11] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, "Evolutionary artificial neural network based on chemical reaction optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2011)*, New Orleans, LA, USA, Jun. 2011.

[12] A. Y. S. Lam and V. O. K. Li, "Chemical reaction optimization: A tutorial," *Memetic Computing*, vol. 4, no. 1, pp. 3–17, Mar. 2012.

[13] M. R. Garey and D. S. Johnsong, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: Freeman & Co Ltd, 1979.

[14] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchii, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: MI: Univ. of Michigan Press, 1975.

[16] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, Nov. 1995.

[17] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment," *Science*, vol. 262, no. 5131, pp. 208–214, 1993.

[18] [Online]. Available: http://www.ncbi.nlm.nih.gov/genbank/

[19] F. H. Previc, "Dopamine and the origins of human intelligence," *Brain and Cognition*, vol. 41, no. 3, pp. 299–350, 1999.