<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>An inter-molecular adaptive collision scheme for Chemical Reaction Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Yu, JJQ; Li, VOK; Lam, AYS</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2014</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/219765">http://hdl.handle.net/10722/219765</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>Congress on Evolutionary Computation (CEC). Copyright © IEEE.; ©2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.; This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
An Inter-molecular Adaptive Collision Scheme for Chemical Reaction Optimization

James J.Q. Yu, Student Member, IEEE and Victor O.K. Li, Fellow, IEEE
Department of Electrical and Electronic Engineering
The University of Hong Kong
Email: {jqyu, vl1}@eee.hku.hk

Albert Y.S. Lam, Member, IEEE
Department of Computer Science
Hong Kong Baptist University
Email: albertlam@ieee.org

Abstract—Optimization techniques are frequently applied in science and engineering research and development. Evolutionary algorithms, as a kind of general-purpose metaheuristic, have been shown to be very effective in solving a wide range of optimization problems. A recently proposed chemical-reaction-inspired metaheuristic, Chemical Reaction Optimization (CRO), has been applied to solve many global optimization problems. However, the functionality of the inter-molecular ineffective collision operator in the canonical CRO design overlaps that of the on-wall ineffective collision operator, which can potential impair the overall performance. In this paper we propose a new inter-molecular ineffective collision operator for CRO for global optimization. To fully utilize our newly proposed operator, we also design a scheme to adapt the algorithm to optimization problems with different search space characteristics. We analyze the performance of our proposed algorithm with a number of widely used benchmark functions. The simulation results indicate that the new algorithm has superior performance over the canonical CRO.

Index Terms—Chemical reaction optimization, global optimization, inter-molecular operator, evolutionary computation, meta-heuristic.

I. INTRODUCTION

Chemical Reaction Optimization (CRO) [1] is a recently proposed population-based general-purpose metaheuristic optimization method. CRO mimics the interactions and transformations of the reactant molecules in chemical reactions to search for global optimums [2]. CRO was designed as a general framework for optimization, and it was initially proposed to solve combinatorial optimization problems [1]. It has been applied to solve a wide range of classical and real-world discrete problems, e.g. Quadratic Assignment Problem [1], Sensor Deployment Problem [3], and Unit Commitment Problem [4]. CRO demonstrated outstanding performance in these problems and was shown to be both effective and efficient in solving similar combinatorial problems.

Based on the CRO optimization framework, Lam et al. proposed a variant of CRO, named Real-Coded Chemical Reaction Optimization (RCCRO), to solve continuous optimization problems [5]. RCCRO has four different operators for the elementary reactions to facilitate the optimization process (the details will be introduced in Section II). The encoding scheme, boundary handling scheme, and other ancillary schemes were also devised. RCCRO has also been applied to solve different benchmark and real-world problems, e.g. training artificial neural networks [6], Optimal Power Flow problem [7], and Cognitive Spectrum Allocation problem [8]. We also conducted research on exploring the nature of different operators in RCCRO. In our previous work, we attempted to use different probability distribution function to replace the Normal distribution adopted to perform neighborhood search [9], and the results show no significant preference over the four distributions studied.

In the canonical design of RCCRO, the inter-molecular ineffective collision operator is largely similar to having two on-wall ineffective collisions occurring simultaneously, and the only difference between inter-molecular and on-wall ineffective collisions is their different energy handling schemes. This design can significantly alleviate the implementation effort as the neighborhood search operator, i.e. on-wall ineffective collision operator, is re-used. However, a potential drawback of this implementation is that the functionalities of these two elementary reactions overlap, which may lead to wasting the limited function evaluations. Moreover, the neighborhood search operator in the canonical RCCRO alters the value of one dimension of the input solution, which can effectively solve problems with minimal inter-dimensional correlation. But it is not efficient at solving problems with strong inter-dimensional correlation such as rotated functions. In order to resolve these drawbacks, this paper focuses on proposing new operators and schemes based on the canonical RCCRO framework.

The rest of this paper is organized as follows. We introduce the canonical design of the CRO framework and the RCCRO design in Section II. The newly proposed inter-molecular ineffective collision operator and the adaptive collision scheme is presented in Section III. Section IV introduces the experimental setting we employed to analyze the performance of our algorithm as well as the simulation results and discussions. Finally this paper is concluded in Section V with some potential future research.

II. CHEMICAL REACTION OPTIMIZATION FRAMEWORK

In this section we will introduce the general framework of the canonical CRO and some implementation details of RCCRO. We will use CRO to refer to both CRO and RCCRO hereafter.
A. Molecule

Molecules are the basic operating agents of CRO. In CRO, each optimization task is considered as a chemical reaction occurring in a closed container with a population of molecules and an energy buffer. The molecules in the container move and collide with the wall or with other molecules. With the collisions, the properties of molecules involved are changed, allowing the algorithm to explore the solution space. Each molecule possesses the following attributes:

- A molecular structure refers to a feasible solution of the problem to be optimized. Each molecule possesses one molecular structure, which is modified by different operators invoked in the elementary reactions.
- Potential Energy (PE) stands for the quality of the solution held by the molecule. In terms of optimization, PE is equivalent to the fitness value of the solution.
- Kinetic Energy (KE) stands for the tolerance of the molecule to accept a worse molecular structure compared to the current one in terms of the solution quality. A larger KE means that the molecule can accept a much worse molecular structure than those molecules with less KE values.

Besides these properties, the molecules can also hold some other attributes to suit different optimization problems. The implementation of these attributes can be problem-independent.

B. Elementary Reactions

In CRO, the optimization task, i.e. the chemical reaction, is tackled by four kinds of elementary reactions, including on-wall ineffective collision (Onwall), decomposition (Dec), intermolecular ineffective collision (Inter), and synthesis (Syn). They occur sequentially and randomly to manipulate the structures of the molecules involved. Among them, Onwall and Dec takes one molecule as input while Inter and Syn takes two molecules. Onwall and Syn output one molecule based on the input molecule(s), while Dec and Inter output two molecules. Thus Dec and Syn can alter the total population size. This dynamic population size property is a feature that distinguishes CRO from other metaheuristics.

1) Neighborhood Search Operator: In the canonical design of CRO, we use this neighborhood search operator as the basic molecular structure manipulating operator [2]. It is employed in Onwall, Dec, and Inter. This operator modifies one molecular structure using Gaussian perturbation value. Assume that the original molecular structure is \( \omega \), and the newly generated molecular structure is \( \omega' \). We first randomly choose a dimension \( i \) of \( \omega \) to modify. Then a random number \( \epsilon \) is generated from a zero-mean Normal distribution whose variance is a user-defined parameter \( \text{stepSize} \). Then the new molecular structure is generated by

\[
\omega'_i = \omega_i + \epsilon
\]

where \( \omega_i \) stands for the \( i \)-th dimension of the molecular structure \( \omega \).

2) Elementary Reaction Operators: Based on the neighborhood search operator, we further designed the four elementary reaction operators. The detailed implementations are elaborated as follows.

- An Onwall occurs when a molecule collides with the container and bounce back. During this elementary reaction, the structure of the involved molecule is manipulated using the neighborhood operator. The main purpose of this elementary reaction is to perform local search.
- A Dec occurs when a molecule collides with the container and breaks into two new molecules. During this elementary reaction, the structure of the input molecule is copied to the two new molecules. Then the new molecules go through a number of independent neighborhood operators. The main purpose of this elementary reaction is to jump out of local optimum.
- An Inter occurs when two molecules collide with each other and bounce back. In the canonical implementation of CRO, this elementary reaction is considered to be two onwalls occurring simultaneously. The main purpose of this elementary reaction is also to perform local search.
- A Syn occurs when two molecules collide and merge into one new molecule. The detailed implementation of this elementary reaction can be found in [2]. The main purpose of this elementary reaction is to maintain the population diversity [9].

C. Searching Pattern

CRO is composed of three phases: initialization, iteration, and the final phase. In initialization, all algorithm parameters are initialized, and the initial population is randomly generated. Then in the iteration phase, the algorithm manipulates the molecules to search for the global optimum in an iterative manner. In each iteration one elementary reaction is selected and conducted. This selection is controlled by the algorithm optimization parameters using a pre-defined deterministic scheme [2]. After the elementary reaction is selected, a corresponding number of molecules are randomly picked to participate in the reaction. These molecules are then manipulated and go through an energy check to determine whether the reaction succeeds or not, i.e. whether the changes are adopted. The energy check is elaborated in [1] and [5], and the underlying concept is the conservation of energy. This phase iterates until the stopping criterion is met. Then the algorithm proceeds to the final phase and the optimum results are output. Interested readers can refer to [1], [2], and [5] for elaboration of the algorithm and its pseudocode.

III. CHEMICAL REACTION OPTIMIZATION WITH ADAPTIVE COLLISION

In this section we will elaborate our proposed intermolecular adaptive collision scheme, which is composed of a new inter-molecular ineffective operator and an adaptive collision scheme. We call this algorithm CRO with Adaptive Collision (CRO/AC). In all discussions hereafter, without loss of generality, we only consider minimization problems.
A. Inter-molecular Ineffective Collision Operator

In the canonical design of CRO, the inter-molecular ineffective collision is considered to be two on-wall ineffective collisions occurring simultaneously. However, this implementation renders the effects of these two operators overlapping. So we propose a new Inter operator to overcome this drawback.

When an Inter takes place, two molecules are randomly selected from the current population to participate in the elementary reaction. Instead of conducting neighborhood operators on them as in canonical CRO, we first check their PE values, i.e. the fitness values of the solutions they hold. We use $\omega_s$ to denote the molecular structure of the molecule with a larger PE value, and $\omega_l$ to denote the one with a smaller PE value. This implies that $\omega_l$ is better than $\omega_s$ in terms of the solution quality.

After the two molecules are examined, we use a two-step approach to modify their structures. In the first step, we change $\omega_s$ to make it more similar to $\omega_l$. This can be accomplished by using the following:

$$\omega^{'s}_{i,j} = (\omega_{t,i,j} - \omega^{'s}_{i,j}) \times r_i + \omega^{'s}_{i,j},$$

where $\omega^{'s}_{i,j}$ stands for the value of $i$-th dimension of $\omega^{'s}_s$, $\omega_{t,i,j}$ stands for the newly generated structure for $\omega_s$, and $r_i$ is a random number uniformly generated in $[0, 1]$. In this step, the random number $r_i$ is independently generated for each dimension.

The second step is to change $\omega_l$ to make it less similar to $\omega^{'s}_s$, accomplished by:

$$\omega^{'l}_{i,j} = (\omega_{t,i,j} - \omega^{'l}_{i,j}) \times r_i + \omega_{t,i,j},$$

The purpose of this design is to avoid pre-mature convergence and maintain the population density. The first step significantly reduces the population density, which may lead to pre-mature convergence. This second step increases the density by a small margin, and acts as a counter-measure of getting stuck in local optimums. We will demonstrate the necessity of this step in Section IV.

The remaining parts of the Inter operator are unchanged. The new molecules will go through the energy conservation test to check whether this Inter is deemed successful or not. If so, the changes made on these molecules, including molecular structures and molecule properties, are accepted. Otherwise these molecules remain unchanged. This ends an Inter elementary reaction for CRO.

B. Adaptive Collision Scheme

One main difference of our proposed Inter operator and the canonical design is that our operator can update the values of multiple dimensions in a solution. This operator is suitable for problems with large inter-dimensional correlations [10], but may have equal or even poorer performance on those problems with no such correlations. Thus the ratio of occurrence of Onwall and Inter is critical to the performance. In the canonical design of CRO, this ratio is generally controlled by a user-defined parameter collision rate $\text{collRate}$ [1], and the value is usually set to be 0.2 [5][11]. However, with the changes in the nature of Inter operator, this value is no longer appropriate and we shall tune the parameter to tailor-make the algorithm for different optimization problems, but this tuning process can be very time-consuming. So we further devise an adaptive collision scheme for CRO to control the ratio of collision adaptively, using the information feedback from the optimization process [12].

In our adaptive collision scheme, we add a new attribute to the system, namely the number of successful inters, denoted counter. Initially, this attribute is set to zero, and at the beginning of each iteration, the value of $\text{collRate}$ is calculated using the following sigmoid function:

$$\text{collRate} = \frac{1}{1 + \exp(-6 \times \frac{\text{counter}}{\text{maxFE}})},$$

where $\text{maxFE}$ is the maximum allowed number of function evaluations. The plot of this function is demonstrated in Fig. 1, where the x-axis is the value of counter and the y-axis is the value of the corresponding $\text{collRate}$. The value of counter is updated whenever an Onwall or Inter occurs and succeeds. If a successful Onwall occurs, the value of counter is reduced by one. Otherwise the value is increased by one. In this way, the algorithm can adjust the $\text{collRate}$ to favor either Onwall or Inter, to match the optimization problem being solved.

C. Other Modifications

In order to fully exploit the utility of our proposed inter-molecular collision scheme, we also make some other modifications to the implementation of the CRO framework.

In the canonical CRO design, the algorithm does not impose any mandatory constraints on the elementary reaction selection until the size of the population is reduced to one. The selection is controlled by the optimization parameters. When the population size is one, neither Inter nor Syn would occur as they need at least two molecules as inputs. However this behavior may potentially reduce the occurrence of Inter and
in return bias the ratio of Onwall and Inter. So we add a mandatory constraint to maintain this ratio.

In CRO/AC, the elementary reaction selection scheme works identically to the canonical design of CRO in the first step, i.e. a random number is first generated and compared with collRate. If the random number is larger, either Onwall or Dec will occur. Otherwise, an Inter or Syn will take place and here we impose the second mandatory condition. If the current population size is no larger than two, all later selection steps are skipped and an Inter will occur. This modification can guarantee that the size of population is always larger than one, and there are always enough molecules in the container to perform an Inter elementary reaction.

IV. EXPERIMENTAL SETTING AND SIMULATION RESULTS

In this section we will first introduce the benchmark functions adopted for performance evaluation and the experimental settings used. The simulation results, comparisons and discussions will also be presented.

A. Benchmark Functions

In order to evaluate our proposed CRO/AC, we conduct a series of simulations on 16 different benchmark functions. These benchmark functions are selected form the benchmark set proposed by Yao et al. [14] and the latest benchmark problem set for the competition on real-parameter single objective optimization at CEC 2013 [13]. The former benchmark set has been adopted for testing performance by a wide range of metaheuristics in recent years [15], and was adopted to test the performance of RCCRO [5]. As revealed in [10], the current benchmark functions often suffer from two major problems: the global optimal points are located at the center of the search space, and the inter-dimensional correlation is weak. So we make a comprehensive test suite to resolve these problems by shifting and rotating some of the benchmark functions. The benchmark functions are listed in Table I. All benchmark functions are optimized in 30 dimensions, and the search spaces are defined to be \([−100, 100]\). The global

<table>
<thead>
<tr>
<th>(f_i)</th>
<th>Function</th>
<th>Solution Transformation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1)</td>
<td>(\sum_{j=1}^{D} \frac{z_j^2}{\sum_{j=1}^{D}</td>
<td>z_j</td>
<td>})</td>
</tr>
<tr>
<td>(f_2)</td>
<td>(\sum_{j=1}^{D} \frac{z_j^2}{\sum_{j=1}^{D}</td>
<td>z_j</td>
<td>})</td>
</tr>
<tr>
<td>(f_3)</td>
<td>(\sum_{j=1}^{D}</td>
<td>z_j</td>
<td>+ \prod_{j=1}^{D}</td>
</tr>
<tr>
<td>(f_4)</td>
<td>(-20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{j=1}^{D} z_j^2}) - \exp\left(\frac{1}{n} \sum_{j=1}^{D} \cos 2\pi z_j\right))</td>
<td>(z = (x - o) \times 0.1)</td>
<td>Shifted Schwefel’s Problem 2.22</td>
</tr>
<tr>
<td>(f_5)</td>
<td>(-20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{j=1}^{D} z_j^2}) - \exp\left(\frac{1}{n} \sum_{j=1}^{D} \cos 2\pi z_j\right))</td>
<td>(z = (x - o) \times 0.32)</td>
<td>Shifted Ackley’s Function</td>
</tr>
<tr>
<td>(f_6)</td>
<td>(-20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{j=1}^{D} z_j^2}) - \exp\left(\frac{1}{n} \sum_{j=1}^{D} \cos 2\pi z_j\right))</td>
<td>(z = (x - o) \times 0.32)</td>
<td>Shifted Ackley’s Function</td>
</tr>
<tr>
<td>(f_7)</td>
<td>(418.9829D - \sum_{j=1}^{D} z_j \sin \sqrt{</td>
<td>z_j</td>
<td>})</td>
</tr>
<tr>
<td>(f_8)</td>
<td>(418.9829D - \sum_{j=1}^{D} z_j \sin \sqrt{</td>
<td>z_j</td>
<td>})</td>
</tr>
<tr>
<td>(f_9)</td>
<td>(\sum_{j=1}^{D} (z_j^2 - 10 \cos 2\pi z_j + 10))</td>
<td>(z = (x - o) \times 0.0512)</td>
<td>Shifted Rastrigin’s Function</td>
</tr>
<tr>
<td>(f_{10})</td>
<td>(\sum_{j=1}^{D} (z_j^2 - 10 \cos 2\pi z_j + 10))</td>
<td>(z = M(x - o) \times 0.0512)</td>
<td>Shifted Rastrigin’s Function</td>
</tr>
<tr>
<td>(f_{11})</td>
<td>(\sum_{j=1}^{D} \frac{z_j^2}{4000} - \prod_{j=1}^{D} \cos \frac{z_j}{4} + 1)</td>
<td>(z = (x - o) \times 6)</td>
<td>Shifted Griewank’s Function</td>
</tr>
<tr>
<td>(f_{12})</td>
<td>(\sum_{j=1}^{D} \frac{z_j^2}{4000} - \prod_{j=1}^{D} \cos \frac{z_j}{4} + 1)</td>
<td>(z = M(x - o) \times 6)</td>
<td>Shifted Griewank’s Function</td>
</tr>
<tr>
<td>(f_{13})</td>
<td>(\sin^2(\pi y_j) + \frac{\sum_{j=1}^{D} (y_j - 1)^2(1 + 10 \sin^2(\pi y_{j+1}))}{\sum_{j=1}^{D} (y_j - 1)^2(1 + 10 \sin^2(\pi y_{j+1}))})</td>
<td>(z = (x - o) \times 0.1)</td>
<td>Shifted Levy’s Function</td>
</tr>
<tr>
<td>(f_{14})</td>
<td>(\frac{1}{4} (y_j - 1)^2(1 + \sin^2(2\pi y_j)), y_j = 1 + \frac{1}{4} (z_j + 11))</td>
<td>(z = (x - o) \times 0.1)</td>
<td>Shifted Levy’s Function</td>
</tr>
<tr>
<td>(f_{15})</td>
<td>(\frac{1}{10} \sin^2(3\pi z_j) + \frac{\sum_{j=1}^{D} (z_j - 1)^2(1 + \sin^2(3\pi z_{j+1}))}{\sum_{j=1}^{D} (z_j - 1)^2(1 + \sin^2(3\pi z_{j+1}))})</td>
<td>(z = (x - o) \times 0.5)</td>
<td>Shifted Penalized Function 1</td>
</tr>
<tr>
<td>(f_{16})</td>
<td>(\frac{1}{10} \sin^2(3\pi z_j) + \frac{\sum_{j=1}^{D} (z_j - 1)^2(1 + \sin^2(3\pi z_{j+1}))}{\sum_{j=1}^{D} (z_j - 1)^2(1 + \sin^2(3\pi z_{j+1}))})</td>
<td>(z = (x - o) \times 0.5)</td>
<td>Shifted Penalized Function 2</td>
</tr>
</tbody>
</table>

* \(o\) is a shifting vector and \(M\) is a transformation matrix. \(o\) and \(M\) can be obtained from [13].

\[ k(x - a)^{m_1} \quad \text{for} \quad x > a \]
\[ 0 \quad \text{for} \quad -a \leq x \leq a \]
\[ k(-x - a)^{m_1} \quad \text{for} \quad x < -a \]
optimum values of these benchmark functions are zero, and all simulation results smaller than $10^{-8}$ are considered to be zero [13].

To evaluate the performance improvement of our proposed CRO/AC over CRO, we compare the simulation results between these two designs. Lam et al. proposed three canonical CRO variants in [5], where the major differences among them are the constraint handling schemes and synthesis operators. We use CRO/BP, CRO/HP, and CRO/BB to refer to RCCR01, RCCR02, and RCCR03 described in [5]. We will compare the performance of CRO/AC with these three algorithms.

CRO/AC and all the CRO variants are implemented in Python 2.7 on Microsoft Windows 7. All simulations are performed on a computer with an Intel Core i7-3770 @ 3.4GHz CPU. In order to reduce statistical errors and generate statistically significant results, each benchmark function is performed on a computer with an Intel Core i7-3770 @ 3.4GHz CPU. In order to reduce statistical errors and generate statistically significant results, each benchmark function is repeated for 51 independent runs for each algorithm according to the suggestion by [13]. We use the maximum number of function evaluation (maxFE) as the termination criteria and maxFE is set to 300 000 for all benchmark functions, satisfying the requirement of [13].

The parameters for all algorithms are set according to the recommendation of [5] for solving multimodal problems, i.e. population size is 20, stepsize is 1, initial energy buffer is $10^5$, initial kinetic energy for molecules is $10^7$, molecular collision rate is 0.2, kinetic energy loss rate is 0.1, decomposition threshold is $1.5 \times 10^5$, and synthesis threshold is 10.

### B. Comparison of CRO/AC and Canonical CRO Variants

The mean values, standard deviations, and Student’s t-test results obtained by CRO/AC and the three canonical CRO variants are presented in Table II. The mean values in bold indicate superiority of the corresponding algorithm over the others. The t-test results are the calculated t-statistic, where a negative value means that CRO/AC outperforms the corresponding algorithm. The t-test values in italic indicate that the advantage is significant at a confidence level of 95%.

From the simulation results the following key points can be observed:

- **CRO/AC** performs better than all canonical CRO variants in most of the tested benchmark functions when comparing the mean simulation results. CRO/AC outperforms the others in 14 functions, and its performance in the remaining two functions ($f_{11}$ and $f_{12}$) are comparable to the CRO variants.

- The t-test results further support our previous observation. CRO/AC outperforms other algorithms in all but three functions ($f_4$, $f_{11}$ and $f_{12}$). In $f_4$, CRO/AC outperforms CRO/HP and performs similarly with CRO/BP and CRO/BB. In the meantime CRO/AC generates more stable simulation results. In $f_{11}$, CRO/AC generates a slightly worse mean result, but the t-test results indicate no significant advantage of the other algorithms.

In terms of the computational time, a very small amount of additional time is required for our inter-molecular adaptive collision scheme. The total extra time is around 2%–7% of the computational time needed by the canonical CRO variants. As the benchmark functions we adopt are relatively less computationally intensive compared with most real-world problems, we believe that the extra time used by CRO/AC is not critical considering the significant performance improvement.

### C. Analysis of the Two-step Inter-molecular Ineffective Collision Operator and Adaptive Collision Scheme

Section III introduced our proposed inter-molecular ineffective collision operator, which is a two-step manipulation method. In this section we will show the necessity of incorporating the second step, i.e. (3). We construct a CRO/AC variant with one-step inter-molecular ineffective collision operator (CRO/AC/1step) which removes the second step of manipulating the input molecules. We also proposed an adaptive collision scheme which adaptively changes the value for the molecular collision rate. We construct CRO/AC/0.2 which

<table>
<thead>
<tr>
<th>Function</th>
<th>CRO/AC</th>
<th>CRO/BD</th>
<th>CRO/BB</th>
<th>CRO/HP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$ Std. Dev.</td>
<td>3.4578e-07</td>
<td>1.5487e-06</td>
<td>1.2265e-06</td>
<td>1.4386e-06</td>
</tr>
<tr>
<td>$f_2$ Std. Dev.</td>
<td>5.100e-06</td>
<td>8.4089e-06</td>
<td>9.3814e-06</td>
<td>9.9651e-06</td>
</tr>
<tr>
<td>$f_3$ Std. Dev.</td>
<td>5.181e-05</td>
<td>3.264e-04</td>
<td>3.972e-04</td>
<td>7.976e-04</td>
</tr>
<tr>
<td>$f_4$ Std. Dev.</td>
<td>6.111e-01</td>
<td>5.733e-01</td>
<td>6.192e-01</td>
<td>6.009e-01</td>
</tr>
<tr>
<td>$f_5$ Std. Dev.</td>
<td>8.047e-01</td>
<td>7.083e-01</td>
<td>6.992e-01</td>
<td>5.511e-01</td>
</tr>
<tr>
<td>$f_6$ Std. Dev.</td>
<td>6.000e-00</td>
<td>5.823e-00</td>
<td>6.120e-00</td>
<td>6.072e-00</td>
</tr>
<tr>
<td>$f_7$ Std. Dev.</td>
<td>6.341e-00</td>
<td>6.176e-00</td>
<td>7.939e-00</td>
<td>6.615e-00</td>
</tr>
<tr>
<td>$f_8$ Std. Dev.</td>
<td>6.000e-00</td>
<td>6.756e-00</td>
<td>7.776e-00</td>
<td>6.497e-00</td>
</tr>
<tr>
<td>$f_9$ Std. Dev.</td>
<td>2.914e-02</td>
<td>7.852e-02</td>
<td>8.403e-02</td>
<td>8.477e-02</td>
</tr>
<tr>
<td>$f_{10}$ Std. Dev.</td>
<td>1.438e-02</td>
<td>4.556e-02</td>
<td>4.182e-02</td>
<td>4.509e-02</td>
</tr>
<tr>
<td>$f_{11}$ Std. Dev.</td>
<td>3.954e-04</td>
<td>8.908e-04</td>
<td>8.829e-04</td>
<td>9.000e-04</td>
</tr>
<tr>
<td>$f_{12}$ Std. Dev.</td>
<td>2.457e-01</td>
<td>2.435e-01</td>
<td>2.457e-01</td>
<td>2.435e-01</td>
</tr>
<tr>
<td>$f_{13}$ Std. Dev.</td>
<td>2.807e-01</td>
<td>2.646e-01</td>
<td>1.139e-01</td>
<td>1.354e-01</td>
</tr>
<tr>
<td>$f_{14}$ Std. Dev.</td>
<td>5.233e-04</td>
<td>3.753e-04</td>
<td>6.931e-04</td>
<td>3.819e-04</td>
</tr>
<tr>
<td>$f_{15}$ Std. Dev.</td>
<td>5.611e-04</td>
<td>4.201e-04</td>
<td>4.395e-04</td>
<td>3.891e-04</td>
</tr>
<tr>
<td>$f_{16}$ Std. Dev.</td>
<td>7.447e-06</td>
<td>1.663e-01</td>
<td>6.926e-01</td>
<td>1.541e-01</td>
</tr>
</tbody>
</table>
sets the collision rate constantly to 0.2 during the entire search process. The same experimental settings and benchmark functions are applied to these algorithms, and the simulation results are presented in Table III. From these test results, we have the following observations:

- Both the two-step manipulation method and the adaptive collision scheme contribute to the performance improvement of CRO/AC. CRO/AC generates 10 best mean values among the 16 benchmark functions while CRO/AC/0.2 produces the best mean values of the other six.

- CRO/AC/0.2 tends to generate outstanding results in the uni-modal optimization functions ($f_1$–$f_4$), while CRO/AC is generally superior at solving multi-modal functions ($f_5$–$f_{10}$).

- The less competitive results generated by CRO/AC/1step are potentially due to pre-mature convergence, which is caused by missing the second step manipulation that allows molecules to jump out of local optima.

The effect of the adaptive collision scheme can also be revealed by Fig. 2, where the changes on $collRate$ of $f_1$ and $f_2$ are plotted. From the figure we can see the adaptive scheme can adjust $collRate$ to different values according to the feedback from the optimization process.

V. CONCLUSION

CRO is a recently proposed simple and powerful metaheuristic optimization method which mimics the interactions and transformations of molecules in chemical reactions to search for the global optimum. CRO has been applied to solve both combinatorial and continuous optimization problems. In the canonical design of CRO for solving continuous optimization problems, the functionalities of on-wall ineffective collision and inter-molecular ineffective collision overlap, and this potentially hampers the performance of CRO. In this paper we propose a novel inter-molecular operator to overcome this drawback. This operator changes the values on multiple dimensions of a solution at one time, and is more efficient at solving problems with strong inter-dimensional correlation than the canonical design. In order to fully utilize the different optimization features of this new operator, we also devise an adaptive collision scheme to control the ratio of different elementary reactions. This scheme learns from the optimization feedback and adjusts the $collRate$ parameter of CRO to adapt the algorithm to different problems. We incorporate the new operator and the adaptive scheme into the CRO framework and propose a new algorithm CRO/AC.

To examine the performance improvement of CRO/AC over canonical CRO, we perform a series of simulations on a wide range of different benchmark functions. The simulation results indicate the superiority of our proposed CRO/AC over all the compared CRO variants. We also analyze the performance improvement contribution made by the new operator and the adaptive scheme, and find both of them are essential for...
CRO/AC to achieve better performance than the canonical CRO.

In the future we will perform a systematic analysis on the parameter selection for CRO/AC. All simulation in this paper adopted the parameter settings devised in [5], which was originally designed for the canonical CRO and may be not suitable for CRO/AC. It is also an interesting topic to combine CRO/AC with the stepSize adaptation scheme proposed in [5]. Last but not least, we will apply CRO/AC to real-world practical optimization problems to perform an overall study of the performance of CRO/AC.

ACKNOWLEDGEMENT

This research is supported in part by the University of Hong Kong Strategic Research Theme on Computation and Information. A.Y.S. Lam is supported in part by the Faculty Research Grant of Hong Kong Baptist University, under Grant No. FRG2/13-14/045.

REFERENCES