Memetic Comp. (2012) 4:3–17
DOI 10.1007/s12293-012-0075-1

REGULAR SURVEY PAPER

Chemical Reaction Optimization: a tutorial

(Invited paper)

Albert Y. S. Lam · Victor O. K. Li

Received: 11 July 2011 / Accepted: 31 January 2012 / Published online: 12 February 2012

Abstract Chemical Reaction Optimization (CRO) is a recently established metaheuristics for optimization, inspired by the nature of chemical reactions. A chemical reaction is a natural process of transforming the unstable substances to the stable ones. In microscopic view, a chemical reaction starts with some unstable molecules with excessive energy. The molecules interact with each other through a sequence of elementary reactions. At the end, they are converted to those with minimum energy to support their existence. This property is embedded in CRO to solve optimization problems. CRO can be applied to tackle problems in both the discrete and continuous domains. We have successfully exploited CRO to solve a broad range of engineering problems, including the quadratic assignment problem, neural network training, multimodal continuous problems, etc. The simulation results demonstrate that CRO has superior performance when compared with other existing optimization algorithms. This tutorial aims to assist the readers in implementing CRO to solve their problems. It also serves as a technical overview of the current development of CRO and provides potential future research directions.

Keywords Chemical Reaction Optimization · Metaheuristic · Nature-inspired algorithm · Approximate algorithm · Optimization

A. Y. S. Lam (✉)
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 273 Cory Hall, Berkeley, CA 94720, USA
e-mail: albertlam@ieee.org

V. O. K. Li
Department of Electrical and Electronic Engineering, The University of Hong Kong, Rm. 610D, Chow Yei Ching Building, Pokfulam Rd., Hong Kong, China
e-mail: vli@eee.hku.hk

1 Introduction

Nature, by itself, is a complex system and human beings and their associated activities are parts of Nature. This complex system operates forever without any problems because there are laws governing the operations of all contained components. The ways in which Nature functions are excellent principles to operate complicated systems and to solve problems. When it comes to mathematics and computer science, these operating methods are called algorithms [29]. Mimicking the behaviors of Nature to achieve goal-oriented activities is called Natured-inspired computing [31]. Thus, we can see that the ideas of imitating phenomena from Nature have great potential in developing algorithms to tackle engineering problems, especially those for which we lack adequate knowledge to design the corresponding efficient solving methods.

Moreover, optimization is one of the cornerstones in engineering and science; most of the problems can be formulated in the form of optimization. It is prevalent, ranging from power generation scheduling in electrical engineering [1], stock market trend prediction in finance [42], to DNA sequencing in biomedical science [32]. Various optimization techniques have been attempted to solve optimization problems. However, it is generally believed that for a class of problems, called nondeterministic polynomial-time hard (NP-hard) in computation complexity theory [11], algorithms obtaining the optimal solutions in polynomial time do not exist [10]. In reality, many important real-world problems are classified as NP-hard. Does it mean that there is no hope to solve these problems efficiently? Auspiciously, we are satisfied with near-optimal (or sub-optimal) solutions most of the time provided that the objective function values of these solutions are not too far away from the optimal. This gives birth to approximate algorithms,

a very influential class of optimization algorithms nowadays.

In the past few decades, the field of Nature-inspired optimization techniques has grown incredibly fast. These algorithms are usually general-purpose and population-based. They are normally referred to as evolutionary algorithms[1] because many of them are motivated by biological evolution. Evolution means "the variation of allele frequencies in populations over time" [2]. We can, without harm, broaden the idea of "evolution" to non-biological processes. In a broad sense, evolutionary algorithms cover those which vary a group of solutions in iterations based on some Nature-inspired operations. Examples include, but are not limited to, Genetic Algorithm (GA) [13], Memetic Algorithm (MA) [6,23], Ant Colony Optimization (ACO) [8], Particle Swarm Optimization (PSO) [15], Differential Evolution (DE) [28], and Harmony Search (HS) [12]. Many of them are inspired by the biological process, varying in scale from the genetic level, e.g. GA, MA, and DE, to the creature level, e.g. ACO and PSO. Unlike the others, HS is motivated by the phenomenon of human activities in composing music.

The aforementioned algorithms are successful in solving many different kinds of optimization problems, as demonstrated by their huge number of citations in the literature. According to the No-Free-Lunch Theorem [35], all metaheuristics which search for extrema are exactly the same in performance when averaged over all possible objective functions. In other words, when one works excellent in a certain class of problems, it will be outperformed by the others in other classes. Therefore, all algorithms which have been shown to address some optimization problems successfully are equally important, as each of them must perform identically well on the average. As the spectrum of optimization problems is huge, the number of reported successful metaheuristics is much less than the number of problems. Recently, Lam and Li [18] proposed a new metaheuristic for optimization, inspired by the nature of chemical reactions. They coined it Chemical Reaction Optimization (CRO). In a short period of time, CRO has been applied to solve many problems successfully, outperforming many existing evolutionary algorithms in most of the test cases. In this tutorial, we introduce this new paradigm, provide guidelines to help the readers implement CRO for their optimization problems, summarize the applications of CRO reported in the literature, and identify possible future research directions for CRO.

The rest of this tutorial is organized as follows. Section 2 gives the inspiration of CRO. We explain the characteristics of CRO in Sect. 3 and introduce the basic elements of CRO in Sect. 4. In Sect. 5, we demonstrate the CRO framework and show how the algorithm is structured. We briefly describe most of the problems addressed by CRO in Sect. 6. We conclude this tutorial and suggest potential future work in Sect. 7.

2 Inspiration

CRO is a technique which loosely couples chemical reactions with optimization. It does not attempt to capture every detail of chemical reactions. In general, the principles of chemical reactions are governed by the first two laws of thermodynamics [14]. Here we explain these laws at a high level to enable the readers to easily grasp the working mechanisms of chemical reactions. The first law (conservation of energy) says that energy cannot be created or destroyed; energy can transform from one form to another and transfer from one entity to another. A chemical reacting system consists of the chemical substances and its surroundings. Each chemical substance possesses potential and kinetic energies, and the energies of the surroundings are symbolically represented by the central energy buffer in CRO.[2] A reaction is endothermic when it requires heat obtained from the surroundings to initialize the reaction process. An exothermic reaction refers to one whose chemical substances give heat to the surroundings. These two kinds of reactions can be characterized by the initial buffer size: when it is positive, the reaction is endothermic; when it is zero, the reaction is exothermic. The second law says that the entropy of a system tends to increase, where entropy is the measure of the degree of disorder. Potential energy is the energy stored in a molecule with respect to its molecular configuration. When it is converted to other forms, the system becomes more disordered. For example, when molecules with more kinetic energy (converted from potential energy) move faster, the system becomes more disordered and its entropy increases. Thus all reacting systems tend to reach a state of equilibrium, whose potential energy drops to a minimum. In CRO, we capture this phenomenon by converting potential energy to kinetic energy and by gradually losing the energy of the chemical molecules to the surroundings.

A chemical system undergoes a chemical reaction when it is unstable, in the sense that it possesses excessive energy. It manipulates itself to release the excessive energy in order to stabilize itself. This manipulation is called chemical reactions. If we look at the chemical substances at the microscopic level, a chemical system consists of molecules, which are the smallest particles of a compound that retain the chemical properties of the compound.[3] Molecules are classified into different species based on the underlying chemical properties. For example, carbon monoxide (CO) and nitrogen

[1] They are also sometimes called metaheuristics.

[2] The central energy buffer will be introduced in Sect. 4.

[3] The definition of molecule is from the General Chemistry Online: Glossary from Frostburg State University, available at http://antoine.frostburg.edu/chem/senese/101/index.shtml.

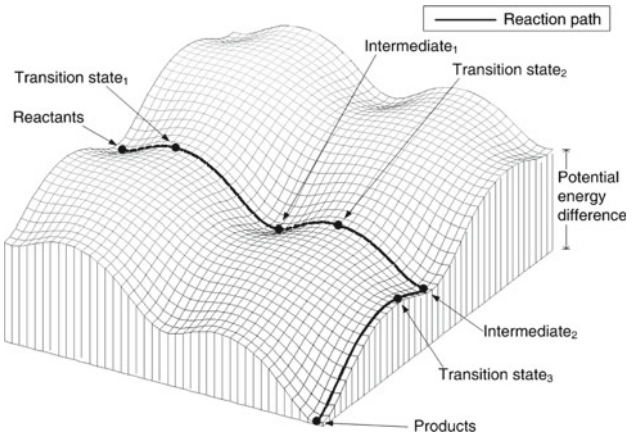

Fig. 1 Illustrative representation of a chemical reaction on the potential energy surface [18]

dioxide (NO_2) are two different chemical species. The chemical system ($CO + NO_2$) is unstable and the unstable chemicals finally convert to more stable species, CO_2 and NO. The overall chemical equation which governs this process can be described by $CO + NO_2 \rightarrow CO_2 + NO$. In fact, this reacting system is realized in multiple stages, described by consecutive sub-reactions[4]: $2NO_2 \rightarrow NO_3 + NO$ and $NO_3 + CO \rightarrow NO_2 + CO_2$. We depict the aforementioned chemical reaction on the potential energy surface in Fig. 1. We can see that a chemical reaction is accomplished by some consecutive (and parallel) sub-reaction steps, from reactants changing to products via intermediates and transition states. A chemical reaction always results in more stable products with minimum energy and it is a step-wise process of searching for the optimal point.

Molecules store energy in the form of chemical bonds; bond formation requires energy from the outside while bond breakage releases energy to the surroundings. A molecule is identified by its molecular structure, which characterizes the contained atoms, bond length, angle, and torsion. Molecules with subtle changes in molecular structures are still considered to belong to the same species.

A chemical change of a molecule is triggered by a collision. There are two types of collisions: uni-molecular and inter-molecular collisions. The former describes the situation when the molecule hits on some external substances (e.g. a wall of the container) while the latter represents the cases where the molecule collides with other molecules. The corresponding reaction change is called an elementary reaction. An ineffective elementary reaction is one which results in a subtle change of molecular structure. We consider four kinds of elementary reactions: on-wall ineffective collision, decomposition, inter-molecular ineffective collision, and synthesis. With respect to molecularity, decomposition and synthesis correspond to on-wall ineffective collision and inter-molecular ineffective collision, respectively, but they have much more vigorous changes to the molecular structures. We summarize the elementary reactions in Table 1.

Table 1 Characteristics of the four elementary reactions

Extent of change	Number of molecules involved	
	Uni-molecular	Inter-molecular
More	Decomposition	Synthesis
Less	On-wall ineffective collision	Inter-molecular ineffective collision

Optimization is the study of problems in which one seeks to minimize (or maximize)[5] a function by systematically choosing the values of the variables in an allowed set. Most of the currently widely utilized optimization algorithms operate iteratively, irrespective of whether they are traditional techniques (e.g. descent methods, Newton's methods [3]) or the evolutionary ones. We can see that the nature of chemical reactions and optimization resemble each other at a high level; they both seek to undergo a series of events step-by-step. Due to the success of other evolutionary algorithms and the No-Free-Lunch Theorem, CRO was proposed based on these observations.

3 Characteristics

Conservation of energy in CRO is as natural selection in GA. Re-distribution of energy among the molecules and interchange of energy from one form to another govern the algorithmic philosophy of CRO. With the underlying assumption of energy conservation and transformation, we manipulate the solutions through a random sequence of elementary reactions. The two ineffective collisions implement local search (intensification) while decomposition and synthesis give the effect of diversification. An appropriate mixture of intensification and diversification makes an effective search of the global minimum in the solution space.

CRO is a variable population-based metaheuristic. The total number of molecules in different iterations may not be the same. When an (on-wall or inter-molecular) ineffective collision happens, the number of molecules before and after the change remains identical. On the other hand, with decomposition and synthesis, the number increases and decreases, respectively. With our definitions of the decomposition and synthesis criteria, we can implicitly influence their frequencies by controlling α and β, respectively (the details will be

[4] Some systems may have the sub-reactions take place spontaneously.

[5] We assume an optimization is a minimization problem. A maximization problem can be converted to the corresponding minimization by adding a minus sign to the objective function.

discussed in Sect. 5). However, the frequencies have a stronger relationship with the "landscape" of the objective function. For example, if "down-hill" search direction is always possible, we will never trigger the decomposition criterion. If "hill-climbing" does not often happen, molecules will not convert their kinetic energy for worse solutions and the synthesis criterion is not invoked. When working with a small set of molecules, we focus on local search more in some regions. Otherwise, we try to spread "seeds" (i.e. molecules) to the whole solution space in a greater extent. Therefore, CRO tries to adapt itself to the problem with the goal of locating the global minimum effectively.

In a broad sense, CRO is an algorithmic framework where we only define the general operations of agents and the energy management scheme. It allows certain implementation details to be adjusted to suit the characteristics of problems. In other words, CRO has high flexibility for users to customize it to meet their own needs.

CRO has a strong relationship with memetic computation, which is defined as "a paradigm that uses the notion of meme(s) as units of information encoded in computational representations for the purpose of problem-solving" [6]. MA hybridizes global and local heuristic search techniques. CRO realizes the global and local search with the elementary reactions. Moreover, we can incorporate (individual and social) learning ability or ideas from other algorithms into CRO through appropriate designs of the decomposition and synthesis mechanisms. This may result in a more powerful CRO-based algorithm for particular problems.

CRO enjoys the advantages of both Simulated Annealing (SA) [17] and GA. The energy conservation requirement gives similar effects of the Metropolis Algorithm used in SA while the decomposition and synthesis operations share similarities with the crossover and mutation operations of GA. When the number of molecules is small, CRO is more like SA. When some crossover and mutation operators are implemented in decomposition and synthesis, CRO performs more like GA (more discussion can be found in Sect. 5.6).

The basic unit in CRO is a molecule. Each molecule has certain attributes, e.g. potential and kinetic energies, molecular structures, etc., with values characterizing that molecule. The elementary reactions define the implementations of molecular interactions. Thus, we can easily program CRO with an object-oriented programming language [30], e.g. C++ and Java. We can define a class whose data fields represent the attributes and whose methods describe the elementary reactions. Whenever a molecule is constructed (in initialization and decomposition), we create an object representing a molecule from the class. If a molecule is removed from the system by combining with another one (in synthesis), we can simply destroy the corresponding object.

Parallelization of CRO can be done without too much effort. When we implement multiple CROs for solving a particular problem, we do not need strong synchronization among the CROs. Unlike other evolutionary algorithms, CRO does not define generations and each iteration involves only a subset of molecules. Each CRO maintains its own population size. Interactions between CROs can be carried out at a certain instant without much restriction as each CRO does not need to wait for another CRO to complete certain actions, e.g. computation of the whole population in a generation in GA. An attempt to parallelize CRO can be found in [39].

To summarize, the advantages of CRO are highlighted as follows:

- CRO is a design framework which allows deploying different operators to suit different problems.
- Its variable population size allows the system to adapt to the problems automatically.
- Conversion of energy and transfer of energy in different entities and in different forms make CRO unique among meterheursitics. CRO has the potential to tackle those problems which have not been successfully solved by other metaheuristics.
- Other attributes can easily be incorporated into the agent (i.e. molecule). This gives flexibility to design different operators.
- CRO enjoys the advantages of both SA and GA.
- CRO can be easily programmed in object-oriented programming language, where a class defines a molecule and methods define the elementary reaction types.
- It is easy to modify CRO to run in parallel, as the population size does not need to be synchronized between computing units.

4 Basic components, elementary reactions, and concepts

In this section, we introduce the building blocks of CRO and explain how they are integrated as a complete algorithm. We first define the manipulated agent, then describe the elementary reactions, and finally elaborate on the core concept of CRO, namely, conservation of energy.

4.1 The manipulated agent

CRO is a multi-agent algorithm and the manipulated agents are molecules. Each molecule has several attributes, some of which are essential to the basic operations of CRO. The essential attributes include (a) the molecular structure (ω); (b) the potential energy (PE); and (c) the kinetic energy (KE). The rest depends on the algorithm operators and they are utilized to construct different CRO variants for particular problems provided that their implementations satisfy the characteristics of the elementary reactions. The optional attributes adopted in most of the published CRO variants are (d) the

number of hits (*NumHit*); (e) the minimum structure (*MinStruct*); (f) the minimum PE (*MinPE*); and (g) the minimum hit number (*MinHit*). Illustrations of the attributes mentioned above are listed in the following:

1. *Molecular structure* ω captures a solution of the problem. It is not required to be in any specific format: it can be a number, a vector, or even a matrix. For example, if the problem solution space is defined as a set of vectors composed of five real numbers, then ω can be any of these vectors.
2. *Potential energy PE* is defined as the objective function value of the corresponding solution represented by ω. If f denotes the objective function, then we have

$$PE_\omega = f(\omega). \tag{1}$$

3. *Kinetic energy KE* is a non-negative number and it quantifies the tolerance of the system accepting a worse solution than the existing one. We will elaborate on the concept later in this section.
4. *Number of hits* When a molecule undergoes a collision, one of the elementary reactions will be triggered and it may experience a change in its molecular structure. *NumHit* is a record of the total number of hits (i.e. collisions) a molecule has taken.
5. *Minimum structure MinStruct* is the ω with the minimum corresponding *PE* which a molecule has attained so far. After a molecule experiences a certain number of collisions, it has undergone many transformations of its structure, with different corresponding *PE*. *MinStruct* is the one with the lowest PE in its own reaction history.
6. *Minimum potential energy* When a molecule attains its *MinStruct*, *MinPE* is the corresponding *PE*.
7. *Minimum hit number MinHit* is the number of hits when a molecule realizes *MinStruct*. It is an abstract notation of time when *Minstruct* is achieved.

4.2 Elementary reactions

There are four types of elementary reactions, each of which takes place in each iteration of CRO. They are employed to manipulate solutions (i.e. explore the solution space) and to redistribute energy among the molecules and the buffer. For demonstration purposes, we will also give examples of the most frequently used operators in various applications of CRO in Sect. 5. Other designs can be found in the references provided in Sect. 6. Note that there is no strict requirements on the mechanisms of the operators and operators designed for other algorithms may also be adopted. However, CRO ensures the conservation of energy when new solutions are generated with the operators.

4.2.1 On-wall ineffective collision

An on-wall ineffective collision represents the situation when a molecule collides with a wall of the container and then bounces away remaining in one single unit. In this collision, we only perturb the existing ω to ω', i.e.,

$$\omega \rightarrow \omega'.$$

This can be done by picking ω' in the neighborhood of ω. Let $N(\cdot)$ be any neighborhood search operator, we have $\omega' = N(\omega)$ and $PE_{\omega'} = f(\omega')$. Moreover, a certain portion of *KE* of the transformed molecule is withdrawn to the central energy buffer (*buffer*). Let *KElossRate* be a parameter of CRO, $0 \leq KELossRate \leq 1$, and $a \in [KELossRate, 1]$ be a random number, uniformly distributed from *KELossRate* to 1. We get

$$KE_{\omega'} = (PE_\omega - PE_{\omega'} + KE_\omega) \times a \tag{2}$$

and the remaining energy, $(PE_\omega - PE_{\omega'} + KE_\omega) \times (1 - a)$, is transferred to *buffer*. If KE_ω is large enough such that the transformed molecule satisfies the following energy conservation condition:

$$PE_\omega + KE_\omega \geq PE_{\omega'} \tag{3}$$

(further discussed in Sect. 4.3), we can have $PE_{\omega'} > PE_\omega$. In other words, we can obtain a worse solution in this elementary reaction. Of course, it is always possible to undergo an on-wall ineffective collision when $PE_{\omega'} \leq PE_\omega$. When a molecule experiences more of this elementary reaction, it will have more *KE* transferred to *buffer*. Hence, the chance of having a worse solution is lower in a subsequent change.

4.2.2 Decomposition

Decomposition refers to the situation when a molecule hits a wall and then breaks into several parts (for simplicity, we consider two parts in our discussion). Assume that ω produces ω'_1 and ω'_2, i.e.,

$$\omega \rightarrow \omega'_1 + \omega'_2.$$

Any mechanism, which can produce ω'_1 and ω'_2 from ω, is allowed. Theoretically, even generating solutions independent of the existing one (random generation of new solution) is feasible. The idea of decomposition is to allow the system to explore other regions of the solution space after enough local search by the ineffective collisions. The effectiveness of the solution generation mechanism is problem-dependent. Since more solutions are created, the total sum of *PE* and *KE* of the original molecule may not be sufficient. In other words, we may have

$$PE_\omega + KE_\omega < PE_{\omega'_1} + PE_{\omega'_2}.$$

As energy conservation is not satisfied, this decomposition has to be aborted. To increase the chance of having a decomposition completed, we randomly draw a small portion of energy from *buffer* to support the change. Let δ_1 and δ_2 be two independent and identically distributed numbers uniformly generated in the range of [0, 1]. We modify the energy conservation condition for decomposition as follows:

$$PE_\omega + KE_\omega + \delta_1 \times \delta_2 \times buffer \geq PE_{\omega'_1} + PE_{\omega'_2}. \quad (4)$$

This models the situation that some energy from *buffer* is transferred to the molecule when it hits the wall. If (4) holds, the existing molecule with ω is replaced by the two newly generated ones, whose *KE*s randomly share the remaining energy $E_{dec} = (PE_\omega + KE_\omega + \delta_1 \times \delta_2 \times buffer) - (PE_{\omega'_1} + PE_{\omega'_2})$, i.e.,

$$KE_{\omega'_1} = E_{dec} \times \delta_3 \quad \text{and} \quad (5)$$

$$KE_{\omega'_2} = E_{dec} \times (1 - \delta_3), \quad (6)$$

where δ_3 is a random number generated in [0, 1]. The energy in the buffer is also updated by

$$buffer' = (1 - \delta_1\delta_2)buffer. \quad (7)$$

4.2.3 Inter-molecular ineffective collision

Inter-molecular ineffective collision takes place when multiple molecules collide with each other and then bounce away. The molecularity (assume two) remains unchanged before and after the process, i.e.,

$$\omega_1 + \omega_2 \rightarrow \omega'_1 + \omega'_2.$$

This elementary reaction is very similar to the uni-molecular ineffective counterpart; we generate ω'_1 and ω'_2 by $\omega'_1 = N(\omega_1)$ and $\omega'_2 = N(\omega_2)$. The energy management is similar but no *buffer* is involved. The energy conservation condition can be stated as

$$PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'_1} + PE_{\omega'_2}. \quad (8)$$

As more molecules are involved, the total sum of energy of the molecular sub-system is larger than that of the on-wall ineffective collision. The probability of the molecules to explore their immediate surroundings is higher. In other words, the molecules have higher flexibility to be transformed to more diverse molecular structures. We can use the same operator for on-wall ineffective collision to produce new solutions. We apply the operator to each molecule to get a new one. If (8) is satisfied, *KE*s of the transformed molecules share the remaining energy $E_{inter} = (PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) - (PE_{\omega'_1} + PE_{\omega'_2})$ in the sub-system, i.e.,

$$KE_{\omega'_1} = E_{inter} \times \delta_4 \quad \text{and} \quad (9)$$

$$KE_{\omega'_2} = E_{inter} \times (1 - \delta_4), \quad (10)$$

where δ_4 is a random number generated in [0, 1].

4.2.4 Synthesis

Synthesis does the opposite of decomposition. A synthesis happens when multiple (assume two) molecules hit against each other and fuse together, i.e.,

$$\omega_1 + \omega_2 \rightarrow \omega'.$$

As only one molecule is produced, it is likely to satisfy the energy conservation condition:

$$PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'}. \quad (11)$$

If (11) holds, the resulting $KE_{\omega'}$ just takes up all the remaining energy, i.e.,

$$KE_{\omega'} = (PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) - (PE_{\omega'}). \quad (12)$$

We can see that we allow greater change to ω' with respect to ω_1 and ω_2 and $KE_{\omega'}$ is usually higher than KE_ω. The resulting molecule has a higher "ability" to explore a new solution region. Any mechanism allowing the combination of solutions is allowed, where the resultant molecule is in a region farther away from the existing ones in the solution space. The idea behind synthesis is diversification of solutions. The implementation detail is again problem-dependent.

4.3 Conservation of energy

One of the fundamental assumptions of CRO is conservation of energy, which means that energy cannot be created or destroyed. The whole system refers to all the defined molecules and the container, which is connected to *buffer*. The total amount of energy of the whole system is determined by the objective function values (i.e. *PE*) of the initial population of molecules whose size is *PopSize*, the initial *KE* (*InitialKE*) assigned, and the initial value of *buffer*. Let $PE_{\omega_i}(t)$, $KE_{\omega_i}(t)$, $PopSize(t)$, and $buffer(t)$ be the *PE* of molecule i, the *KE* of molecule i, the number of molecules, and the energy in the central buffer at time t. When the algorithm evolves, the total amount of energy in the system always remains constant, i.e.,

$$\sum_{i=1}^{PopSize(t)} (PE_{\omega_i}(t) + KE_{\omega_i}(t)) + buffer(t) = C, \quad (13)$$

where C is a constant. Each elementary reaction manages a sub-system (i.e. a subset of entities of the system); a uni-molecular collision involves a molecule and the container while an inter-molecular collision concerns multiple molecules. After an elementary reaction, the total energy of the constructed sub-system remains the same. Let k and l be the

number of molecules involved before and after a particular elementary reaction, and let ω and ω' be the molecular structures of an existing molecule and the one to be generated from the elementary reaction, respectively. In general, the elementary reaction can only take place when it satisfies the following energy conservation condition:

$$\sum_{i=1}^{k} (PE_{\omega_i} + KE_{\omega_i}) \geq \sum_{i=1}^{l} PE_{\omega_i'}. \quad (14)$$

We modify this condition for decomposition as it involves *buffer* on the left-hand side of (14). Note that *PE* is determined by (1) according to the molecular structure. If the resultant molecules have very high potential energy, i.e. they give very bad solutions, the reaction will not occur.

Theoretically, energy cannot attain a negative value and any operation resulting in negative energy should be forbidden. However, some problems may attain negative objective function values (i.e. negative *PE*), but we can convert the problem to an equivalent one by adding an offset to the objective function to make each *PE* non-negative. The law of conservation of energy is still obeyed and the system works perfectly. Interested readers may refer to [20] for more details.

5 Algorithm design

In this section, we will guide the readers to develop a basic version of CRO. This serves to help the readers understand how CRO works. We also give examples of some common operators used in CRO. Although CRO is a general-purpose metaheuristic, as with other general-purpose metaheuristics, this basic CRO may not give good performance to all problems of interest. At the end of this section, we also give some suggestions on how to proceed to more advanced versions of CRO which can be more adaptive to the problem.

Similar to other evolutionary algorithms or metaheuristics, CRO consists of three stages: initialization, iterations, and the final stage. We define the elements of the algorithms in the initialization and the algorithm explores the solution space in iterations. In the final stage, the algorithm terminates and the best found solution is output.

5.1 Initialization

Imagine there is a container with some chemical substances inside in the forms of molecules. We initialize the settings of the algorithm and assign values to the algorithmic parameters, including *PopSize*, *KELossRate*, *MoleColl*, *buffer*, *InitialKE*, α, and β.[6]

[6] We will discuss *MoleColl*, α, and β in Sect. 5.2.

Algorithm 1 "Molecule" class

```
1: class Molecule
2:   Attributes:
3:     ω, PE, KE, NumHit, MinStruct, MinPE, MinHit
4:   Method:
5:     Molecule() \\constructor
6:     {
7:       Randomly generate ω in the solution space
8:       PE ← f(ω)
9:       KE ← InitialKE
10:      NumHit ← 0
11:      MinStruct ← ω
12:      MinPE ← PE
13:      MinHit ← 0
14:    }
15:    Onwall IneffectiveCollision()
16:    Decomposition()
17:    IntermolecularIneffectiveCollision()
18:    Synthesis()
19: end class
```

In this stage, we define the manipulated agent, i.e. a molecule, set the parameter values, and construct the initial population of molecules. As mentioned in Sect. 3, it is preferred to program CRO with an object-oriented programming language [30]. We create a "Molecule" class with some attributes and methods. The attributes are those mentioned in Sect. 4.1 while we define five methods in the class, including the class constructor and the four elementary reactions.[7] The constructor defines the details of an object when it is created according to the class. Here the object refers to a "molecule". As we normally generate the initial set of solutions randomly in the solution space, we assign a random solution to ω in the constructor. The pseudocode of the "Molecule" class is given in Algorithm 1. We create *PopSize* number of molecules from "Molecule" to form the initial population of molecules.

5.2 Iterations

Molecules with energy move and trigger collisions. A molecule can either hit on a wall of the container or collide with each other. This is decided by generating a random number b in [0, 1]. If $b >$ *MoleColl* or the system only has one molecule, we have a uni-molecular collision. Otherwise, an inter-molecular collision follows.

For a uni-molecular collision, we randomly select one molecule from the population and decide if it results in an on-wall ineffective collision or a decomposition, by checking the decomposition criterion on the chosen molecule. In most of the CRO applications in Sect. 6, the decomposition

[7] As we only carry out the elementary reactions in the iterations stage, we will explain their implementations in Sect. 5.2.

criterion is defined as

$$NumHit - MinHit > \alpha. \tag{15}$$

This means that the molecule has undergone α times of local search without locating a better local minimum. Thus we should explore other parts of the solution space through decomposition. Other definitions of decomposition criteria are allowed provided that diversification takes place suitably in between intensifications. If (15) is satisfied, it will result in a decomposition. Otherwise, we get an on-wall ineffective collision. The pseudocodes of on-wall ineffective collision and decomposition are shown in Algorithms 2 and 3, respectively.

Algorithm 2 OnwallIneffectiveCollision

Input: molecule M_ω
$\omega' \leftarrow N(\omega)$
$PE_{\omega'} \leftarrow f(\omega')$
$NumHit_\omega \leftarrow NumHit_\omega + 1$
if $PE_\omega + KE_\omega \geq PE_{\omega'}$ **then**
  Generate $a \in [KELossRate, 1]$
  $KE_{\omega'} \leftarrow (PE_\omega - PE_{\omega'} + KE_\omega) \times a$
  $buffer \leftarrow buffer + (PE_\omega - PE_{\omega'} + KE_\omega) \times (1 - a)$
  $\omega \leftarrow \omega'$
  $PE_\omega \leftarrow PE_{\omega'}$
  $KE_\omega \leftarrow KE_{\omega'}$
  if $PE_\omega < MinPE_\omega$ **then**
    $MinStruct_\omega \leftarrow \omega$
    $MinPE_\omega \leftarrow PE_\omega$
    $MinHit_\omega \leftarrow NumHit_\omega$
  end if
end if

Algorithm 3 Decomposition

Input: molecule M_ω
Create $M_{\omega'_1}$ and $M_{\omega'_2}$
Obtain ω'_1 and ω'_2 from ω
$PE_{\omega'_1} \leftarrow f(\omega'_1)$ and $PE_{\omega'_2} \leftarrow f(\omega'_2)$
if $PE_\omega + KE_\omega \geq PE_{\omega'_1} + PE_{\omega'_2}$ **then**
  $E_{dec} \leftarrow PE_\omega + KE_\omega - (PE_{\omega'_1} + PE_{\omega'_2})$
  goto Step 13
else
  Generate $\delta_1, \delta_2 \in [0, 1]$
  $E_{dec} \leftarrow PE_\omega + KE_\omega + \delta_1\delta_2 \times buffer - (PE_{\omega'_1} + PE_{\omega'_2})$
  if $E_{dec} \geq 0$ **then**
    $buffer \leftarrow buffer \times (1 - \delta_1\delta_2)$
    Generate $\delta_3 \in [0, 1]$
    $KE_{\omega'_1} \leftarrow E_{dec} \times \delta_3$ and $KE_{\omega'_2} \leftarrow E_{dec} \times (1 - \delta_3)$
    $MinStruct_{\omega'_1} \leftarrow \omega'_1$ and $MinStruct_{\omega'_2} \leftarrow \omega'_2$
    $MinPE_{\omega'_1} \leftarrow PE_{\omega'_1}$ and $MinPE_{\omega'_2} \leftarrow PE_{\omega'_2}$
    Destroy M_ω
  else
    $NumHit_\omega \leftarrow NumHit_\omega + 1$
    Destroy $M_{\omega'_1}$ and $M_{\omega'_2}$
  end if
end if

Similarly, for an inter-molecular collision, we randomly select two molecules from the population and determine if there will be an inter-molecular ineffective collision or a synthesis by checking the synthesis criterion on the chosen molecules. We usually adopt the following definition in the deployments of CRO in Sect. 6: all involved molecules satisfy

$$KE \leq \beta. \tag{16}$$

That means all involved molecules have kinetic energy less than or equal to β. Molecules with too low KE lose the flexibility of escaping from local minima. We trigger a synthesis to bring those inflexible molecules to other solution regions for exploration. If (16) for each involved molecule is satisfied, it will result in a synthesis. Otherwise, we have an inter-molecular ineffective collision. The pseudocodes of inter-molecular ineffective collision and synthesis are shown in Algorithms 4 and 5, respectively.

Algorithm 4 IntermolecularIneffectiveCollision

Input: molecules M_{ω_1} and M_{ω_2}
$\omega'_1 \leftarrow N(\omega_1)$ and $\omega'_2 \leftarrow N(\omega_2)$
$PE_{\omega'_1} \leftarrow f(\omega'_1)$ and $PE_{\omega'_2} \leftarrow f(\omega'_2)$
$NumHit_{\omega_1} \leftarrow NumHit_{\omega_1} + 1$ and $NumHit_{\omega_2} \leftarrow NumHit_{\omega_2} + 1$
$E_{inter} \leftarrow (PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) - (PE_{\omega'_1} + PE_{\omega'_2})$
if $E_{inter} \geq 0$ **then**
  Generate $\delta_4 \in [0, 1]$
  $KE_{\omega'_1} \leftarrow E_{inter} \times \delta_4$ and $KE_{\omega'_2} \leftarrow E_{inter} \times (1 - \delta_4)$
  $\omega_1 \leftarrow N(\omega'_1)$ and $\omega_2 \leftarrow N(\omega'_2)$
  $PE_{\omega_1} \leftarrow PE_{\omega'_1}$ and $PE_{\omega_2} \leftarrow PE_{\omega'_2}$
  $KE_{\omega_1} \leftarrow KE_{\omega'_1}$ and $KE_{\omega_2} \leftarrow KE_{\omega'_2}$
  if $PE_{\omega_1} < MinPE_{\omega_1}$ **then**
    $MinStruct_{\omega_1} \leftarrow \omega_1$
    $MinPE_{\omega_1} \leftarrow PE_{\omega_1}$
    $MinHit_{\omega_1} \leftarrow NumHit_{\omega_1}$
  end if
  if $PE_{\omega_2} < MinPE_{\omega_2}$ **then**
    $MinStruct_{\omega_2} \leftarrow \omega_2$
    $MinPE_{\omega_2} \leftarrow PE_{\omega_2}$
    $MinHit_{\omega_2} \leftarrow NumHit_{\omega_2}$
  end if
end if

Algorithm 5 Synthesis

Input: molecules M_{ω_1} and M_{ω_2}
Create $M_{\omega'}$
Obtain ω' from ω_1 and ω_2
$PE_{\omega'} \leftarrow f(\omega')$
if $PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'}$ **then**
  $KE_{\omega'} \leftarrow (PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2}) - PE_{\omega'}$
  $MinStruct_{\omega'} \leftarrow \omega'$
  $MinPE_{\omega'_1} \leftarrow PE_{\omega'}$
  Destroy M_{ω_1} and M_{ω_2}
else
  $NumHit_{\omega_1} \leftarrow NumHit_{\omega_1} + 1$ and $NumHit_{\omega_2} \leftarrow NumHit_{\omega_2} + 1$
  Destroy $M_{\omega'}$
end if

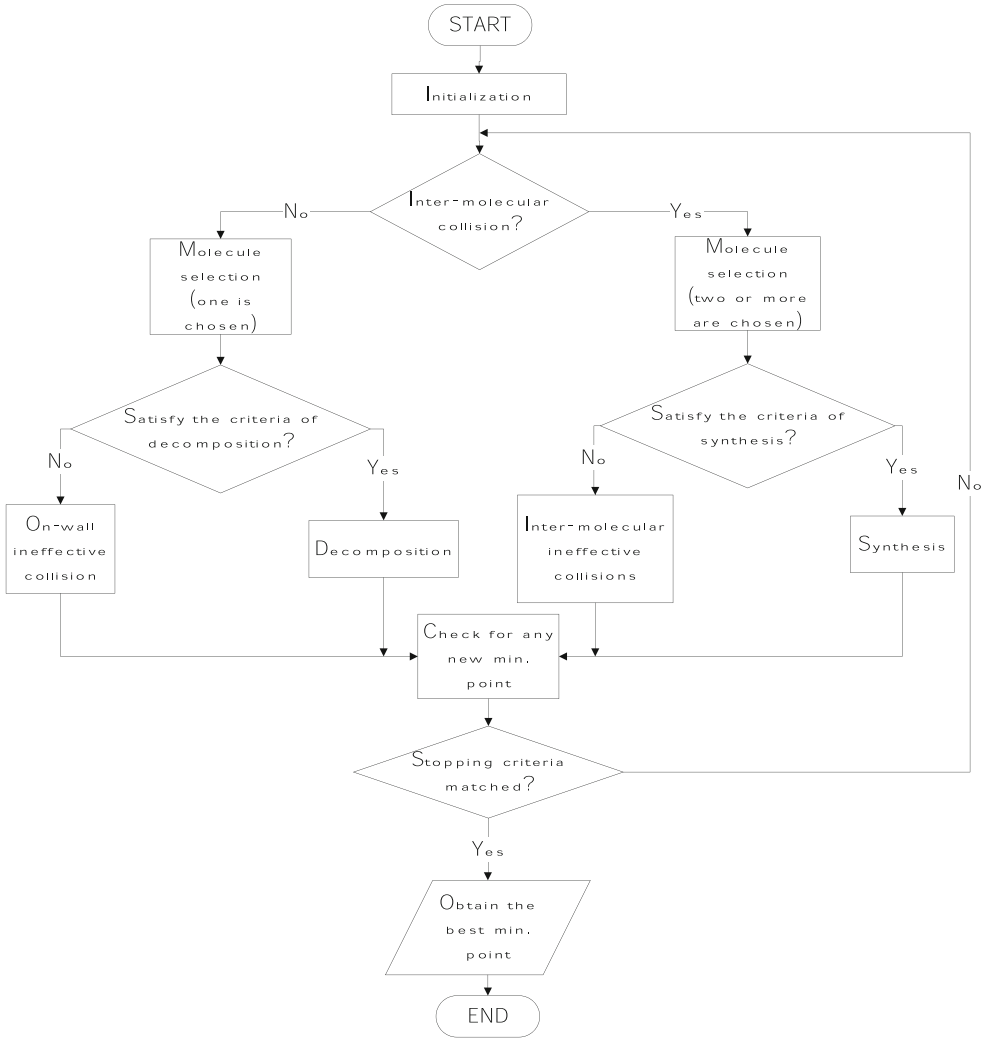

Fig. 2 Schematic diagram of CRO [18]

Inequalities (15) and (16) control the degree of diversification by α and β. Proper values of α and β balance intensification (i.e. exploitation) and diversification (i.e. exploration).

After an elementary reaction (manipulation of solutions) completes, we check if the energy conservation condition is obeyed. If not, the change is abolished. Then we check if any newly determined solution has a lower objective function value. If so, we record the best solution obtained so far. If no stopping criteria are met, we will start a new iteration.

5.3 The final stage

If any of the stopping criteria is met, we will go to the final stage. The stopping criteria are defined according to the user's requirements and preferences. Typical stopping criteria include the maximum amount of CPU time used, the maximum number of function evaluations performed, obtaining an objective function value less than a predefined threshold, the maximum number of iterations performed without improvements, etc. In this stage, we simply output the best solution found with its objective function value and terminate the algorithm.

5.4 The overall algorithm

To program CRO, we just need to assemble the previously mentioned components together according to the peudocode given in Algorithm 6. To ease understanding the flow of the algorithm, we also give the schematic diagram of CRO in Fig. 2.

Here are some suggested values for the parameters: $PopSize = 10$, $KELossRate = 0.2$, $MoleColl = 0.2$, $InitialKE = 1000$, $\alpha = 500$, $\beta = 10$, and $buffer = 0$. These values are deduced from our implementation in [18,20]. However, these parameter values are problem-dependent. To maximize the performance of CRO for a particular problem, the readers may perform some parameter tunings to determine a good combination of parameter values.

Algorithm 6 CRO

1: **Input:** Objective function f and the parameter values
2: \\ Initialization
3: Set *PopSize*, *KELossRate*, *MoleColl*, *buffer*, *InitialKE*, α, and β
4: Create *PopSize* number of molecules
5: \\ Iterations
6: **while** the stopping criteria not met **do**
7: Generate $b \in [0, 1]$
8: **if** $b >$ *MoleColl* **then**
9: Randomly select one molecule M_ω
10: **if** Decomposition criterion (15) met **then**
11: Trigger *Decomposition*
12: **else**
13: Trigger *OnwallIneffectiveCollision*
14: **end if**
15: **else**
16: Randomly select two molecules M_{ω_1} and M_{ω_2}
17: **if** Synthesis criterion (16) met **then**
18: Trigger *Synthesis*
19: **else**
20: Trigger *IntermolecularIneffectiveCollision*
21: **end if**
22: **end if**
23: Check for any new minimum solution
24: **end while**
25: \\ The final stage
26: Output the best solution found and its objective function value

5.5 Operator examples

Here we give examples of operators used in some applications of CRO given in Sect. 6.

5.5.1 Two-exchange

It is also called pair-exchange or 2-opt [5]. It is a neighborhood search operator $N(\cdot)$ for combinatorial problems. It can be used in Line 2 of Algorithm 2 and Line 2 of Algorithm 4.

Consider a problem with solutions in the form of vectors of n elements. Let $\omega = [\omega(i), 1 \le i \le n]$ be a particular solution. First we randomly pick two distinct elements from ω, e.g., $\omega(i)$ and $\omega(j)$, where $i < j$. Then we form a new solution ω' by exchanging their positions, i.e., $\omega' = [\omega(1), \ldots, \omega(i-1), \omega(j), \omega(i+1), \ldots, \omega(j-1), \omega(i), \omega(j+1), \ldots, \omega(n)]$. If the problem is confined to a permutation vector space, this operator can guarantee that ω' is still a permutation vector as long as ω is a permutation vector.

5.5.2 Gaussian perturbation with reflection

Gaussian perturbation is also called Gaussian mutation [4]. It is a neighborhood search operator $N(\cdot)$ for continuous problems. It can be used in Line 2 of Algorithm 2 and Line 2 of Algorithm 4.

Consider a problem with continuous solution space. Let $\omega = [\omega(i), 1 \le i \le n]$ where $\omega(i) \in [l_i, u_i], l_i \le u_i; l_i, u_i \in \mathcal{R}, \forall i$. First we randomly pick an element $\omega(i)$ from ω. Let Δ_i be a random variable with a Gaussian probability density function having zero mean and variance σ^2. Let δ_i be a realization of Δ_i. We have $\tilde{\omega}(i) = \omega(i) + \delta_i$. If $\tilde{\omega}(i)$ is smaller than l_i, we get $\omega'(i)$ by reflecting on l_i with the amount of violation. Otherwise, we have $\tilde{\omega}(i) = \omega(i)$. If $\tilde{\omega}(i)$ is larger than u_i, we obtain $\omega'(i)$ similarly by reflecting on u_i. Mathematically, we get $\omega'(i)$ by

$$\omega'(i) = \begin{cases} 2l_i - \tilde{\omega}(i) & \text{if } \tilde{\omega}(i) < l_i, \\ 2u_i - \tilde{\omega}(i) & \text{if } \tilde{\omega}(i) > u_i, \\ \tilde{\omega}(i) & \text{otherwise.} \end{cases} \tag{17}$$

5.5.3 Half-total change

It is an example of a decomposition operator and it can be used in Line 3 of Algorithm 3. As its name implies, we produce a new solution from an existing one by keeping one half of the existing solution values and assigning the remaining half with new values. Suppose we try to produce two new solutions $\omega'_1 = [\omega'_1(i), 1 \le i \le n]$ and $\omega'_2 = [\omega'_2(i), 1 \le i \le n]$ from $\omega = [\omega(i), 1 \le i \le n]$. For ω'_1, we first copy ω to ω'_1 and then randomly pick $\lfloor n/2 \rfloor$ elements in the vector of ω'_1, where $\lfloor \cdot \rfloor$ returns the largest integer not greater than the argument. For each of these elements, e.g., $\omega'_1(i)$, we assign a new value according to the problem constraints. For example, if $\omega'_1(i)$ can only take a value in a set S_i, we can just randomly select an element from S_i to $\omega'_1(i)$. If S_i is a continuous set, we can add a random perturbation to it to get a new $\omega'_1(i)$, similar to the Gaussian perturbation with reflection scheme mentioned in the previous subsection for the ith element of ω'_1. After producing ω'_1, we produce ω'_2 similarly. As the randomly chosen elements of ω'_1 and ω'_2 and the newly assigned values are different, ω'_1 is quite different from ω'_1, and also from ω.

5.5.4 Probabilistic select

It is an example of synthesis operator and it can be used in Line 3 of Algorithm 5. Suppose we produce solution $\omega' = [\omega'(i), 1 \le i \le n]$ by combining $\omega_1 = [\omega_1(i), 1 \le i \le n]$ and $\omega_2 = [\omega_2(i), 1 \le i \le n]$. This operator tries to randomly select elements from ω_1 and ω_2 to form ω'. To do this, we assign each $\omega'(i)$ with a value equal to either $\omega_1(i)$ or $\omega_2(i)$ randomly.

5.6 Advanced settings

We have only specified how the basic CRO works. Due to the No-Free-Lunch Theorem [35], it cannot have good performance for all kinds of problems. Recall that CRO is cast as a general-purpose algorithm and it is stated in the form

of an algorithmic framework. Many details can be modified to suit a particular problem. Here we attempt to give the readers some directions for developing advanced versions.

In the basic CRO, we always specify the maximum number of molecules involved in an elementary reaction to be two. Each elementary reaction needs to satisfy the energy conservation condition (14) in order to realize a new solution. However, if more than two molecules are involved (except the on-wall ineffective collision which is always one-to-one), more energy may be committed and the molecules may attain new solutions to a greater extent; more molecules may compensate others for a great change in *PE*. For example, we may allow an inter-molecular ineffective collision with three molecules: $\omega_1 + \omega_2 + \omega_3 \rightarrow \omega_1' + \omega_2' + \omega_3'$.

We only give the principles of the elementary reactions in Sect 5.2, where operators are required to specify how to generate new solutions from existing ones. In Sect. 5.5, we give examples of some commonly used operators of CRO. Similar to other evolutionary algorithms, it is possible to design good operators to gain better performance for a particular problem. Moreover, we can also adopt the operators successfully used in other algorithms in CRO. For example, the effect of decomposition is similar to that of mutation in GA. We can apply a mutation operator to a molecule twice to produce two different molecules. We can also apply a GA crossover operator in synthesis to combine solutions into one.

We can design the decomposition and synthesis criteria other than those given in (15) and (16). Decomposition and synthesis bring diversification to the algorithm. Diversification cannot take place too often, or it will be become a completely random algorithm. The criteria specify when a diversification happens in between intensifications. Recall that in Sect. 4.1, only the molecular structure ω, *PE*, and *KE* are necessary to characterize a molecule. Other attributes are optionally used to describe the condition of a molecule for checking the decomposition and synthesis criteria. Other attributes can also be introduced for different designs of the criteria.

Many optimization problems impose constraints to differentiate feasible solutions from the infeasible ones. New solutions generated in the elementary reactions may be feasible or infeasible depending on the operators used. There are generally three approaches to handle constraints:

1. One can design the operators which always map to feasible solutions.
2. We allow the operators to produce infeasible solutions but we introduce a mechanism to convert any infeasible solutions into feasible ones at the end of each iteration.
3. We allow infeasible solutions without any correction mechanism but we impose a penalty at the objective function to any infeasible solution.

More information about constraint-handling techniques can be found in [9].

6 Applications

Although CRO is a newly proposed algorithm, it has been applied to problems in many disciplines successfully. CRO has been compared with many existing evolutionary approaches and it achieves very competitive or even superior performance. Applications of CRO with the operators used are summarized in Table 2.

6.1 Quadratic assignment problem

Quadratic Assignment Problem is a fundamental combinatorial problem in operations research [22]. It belongs to location analysis, about minimizing the transportation cost by varying the locations of facilities. Consider the assignment of n facilities to n locations. We know the distance between each pair of locations and the human flow between each pair of facilities. The problem is to minimize the total cost (distance $\times$ flow) by arranging the locations of the facilities. In [18], the earliest version of CRO is compared with the variants of some popular evolutionary algorithms and CRO achieves superior performance in many test instances. In [39], a parallel version of CRO with a synchronous communication strategy is proposed to tackle the problem. The computation time and the solution quality of the parallel implementation are improved when compared to those of the sequential version of CRO.

6.2 Resource-constrained project scheduling problem

Resource-Constrained Project Scheduling Problem is one of the most intractable, NP-hard optimization problems in operations research, related to project management, resource allocation, and the manufacturing process [7]. Consider that time is divided into slots and there are some activities to be scheduled for a project. Each activity requires resources to process and it may span more than one time slot. Resources are limited; we need to decide which activities should be supported in a certain time slot. There are also precedence constraints among the activities. In other words, some activities can only start when certain ones have been completed. The objective is to minimize the lifespan of the project. In [18], CRO can achieve the known global minimums of most instances in the standard benchmarks.

6.3 Channel assignment problem in wireless mesh networks

A wireless mesh network is composed of some stationary wireless mesh routers, each of which is equipped with certain

Table 2 Applications of CRO

Problem	Ref.	Type	Field	Year	Solution Structure	Neighborhood operator	Decomposition operator	Synthesis operator
Quadratic Assignment Problem	[18,39]	Combinatorial, NP-hard	Operations research	2010	Permutation vector	Two-exchange	Circular shift	Distance-preserving crossover
Resource-Constrained Project Scheduling Problem	[18]	Combinatorial, NP-hard	Operations research	2010	Permutation vector	Two-exchange	Circular shift	Distance-preserving crossover
Channel Assignment Problem in wireless mesh networks	[18]	Combinatorial, NP-hard	Communications, Networking	2010	Integer vector	One-difference	Half-total-change	Probabilistic select
Population Transition Problem in peer-to-peer live streaming	[19]	Continuous	Communications, Networking	2010	Right stochastic matrix	Randomly redistribute the sum of two random numbers into two	Randomly assign rows to new solutions with random generation of unassigned rows	Probabilistic select on rows
Cognitive Radio Spectrum Allocation Problem	[21]	Combinatorial, NP-hard	Communications, Networking	2010	Binary vector	One-difference	Randomly assign bits to new solutions with random generation of unassigned bits	Probabilistic select
Grid Scheduling Problem	[36,37]	Combinatorial, NP-hard	Computing	2010, 2011	Permutation vector, integer vector	Insertion, two-exchange, One-difference	Random generation, half-random	Position-based, one-position exchange
Standard continuous benchmark functions	[20]	Continuous	Mathematics	2011	Real vector	Gaussian perturbation	Half-total-change	Probabilistic select, BLX-α
Stock Portfolio Selection Problem	[38]	Mixed-integer, multi-objective, NP-hard	Finance	2011	Mixed-integer vector	One-difference	Half-random	Keep the aligned numbers and randomly generate the rest
Artificial neural network training	[41]	Continuous	Computational intelligence	2011	Real matrices and vectors	Gaussian perturbation	Perturb every element with 0.5 probability	Probabilistic select
Network Coding Optimization Problem	[25]	Combinatorial, NP-hard	Communications, Networking	2011	Integer vector	One-difference	Randomly generate a solution and modify the coding links	Probabilistic select

radio interfaces. Two routers establish a communication link if they are located in the transmission range of each other with the same channel assigned to one of their interfaces. There are only limited channels available and two established communication links on the same channel interfere each other if they are in close proximity. The Channel Assignment Problem assigns channels to the communications links so as to minimize the induced interference subject to interface constraint, which means that we cannot assign more channels to a router than the number of interfaces equipped. This problem is NP-hard and combinatorial [33]. CRO can improve the existing solutions to the problem [18].

6.4 Population Transition Problem in peer-to-peer live streaming

In a peer-to-peer live streaming system, there is a stream source providing streaming data, together with peers receiving the data. Due to heterogeneous network conditions, peers experience different transmission delays for the data from the source and they can be grouped into colonies according to the delays. For a particular peer, its upstream peers with shorter transmission delays and those in the same colony can serve as a source for the data. The system is in universal streaming when all peers are served with sufficient streaming data. Peers can join and leave the system and switch to another colony while the system can impose rules to guide peers to join the colonies (e.g. assign transition probability for peers transiting from colony to colony). Population Transition Problem maximizes the probability of universal streaming by assigning population transition probabilities among all colonies [19]. In [19], CRO is compared with some practical strategies and simulation shows that the evolutionary approach by CRO performs better than the non-evolutionary ones.

6.5 Cognitive radio spectrum allocation problem

In many countries, wireless channel utilization is regulated and most of the channels can only be used by authorized users. Due to the widespread employment of wireless devices, the shared channels become overcrowded and their quality of service deteriorates. With underutilization of the restricted channels, the capacity of the whole wireless system will substantially increase when unauthorized users are allowed to use the restricted channels provided that higher priority is given to the authorized users. Adjoining users on the same channel induce interference. Restricted to an interference-free environment, the spectrum allocation problem assigns channels to users in order to maximize system utility subject to hardware constraint (which is similar to the interface constraint in Sect. 6.3) [26]. CRO shows dramatic improvement over other existing approaches [21].

6.6 Grid scheduling problem

Grid computing is the next wave of computing where we delegate computational tasks to the computer cloud (e.g. Internet), instead of on a standalone machine [27]. The tasks are taken up by idle computing resources and computed results are then returned to the requester. The resources may be heterogeneous in computational power and volatile. Grid Scheduling Problem schedules tasks to resources so as to minimize computational overheads and to utilize the resources effectively. Several variants of CRO are proposed with different considerations of solution representation and priority of the elementary reactions [36,37]. CRO outperforms many existing evolutionary methods in most test cases.

6.7 Standard continuous benchmark functions

Many optimization problems are continuous problems. The original CRO [18] mainly focuses on discrete problems and a successful general-purpose metaheuristic should also be applicable to the continuous domain. CRO is extended to solve continuous problems systematically in [20]. This continuous version is tested with the standard benchmarks comprised of unimodal, high-dimensional and low-dimensional multimodal functions [40]. Many existing evolutionary approaches are compared with CRO, which show very competitive results. An adaptive scheme for CRO is also proposed in [20].

6.8 Stock portfolio selection problem

Investing in a selection of stocks, instead of a single one, is almost the golden rule in finance to reduce risk. There is always a tradeoff in investment: minimizing the risk while maximizing the return. Stock portfolio selection studies how to build a portfolio of stocks with the consideration of the two contradicting objectives [34]. It is a multi-objective mixed-integer NP-hard problem. In [38], CRO is employed to solve the problem based on the Markowitz model and the Sharpe ratio. A super molecule-based CRO is proposed to compute the Pareto frontier with better performance in terms of Sharpe ratio, expected return, and variance than the canonical form.

6.9 Artificial neural network training

An artificial neural network is a very successful tool to model the input–output relationships of complex systems. The network is formed by interconnecting artificial neurons arranged in layers, simulating a biological neural network [24]. It is an adaptive system with the ability to learn from provided data. It has many real-world applications, e.g. classification, function approximation, and data mining. In order to model a system, a neural network requires a set of related data for

training, i.e., to evolve the network structure and to tune the weights. In [41], CRO is employed to train neural networks. The CRO-trained neural networks have the best testing error rate among many representative evolutionary schemes.

6.10 Network coding optimization problem

In a traditional computer networks, sources send data to destinations via some routers and the routers only receive and forward the data without further processing. Network coding enhances network performance when routers are endowed with processing ability (coding). Network coding can increase the system throughput without any topological changes to the network. However, enabling coding on all possible links increases computational cost. Network Coding Optimization Problem minimizes the number of coding links while maintaining a certain transmission rate [16]. It is an NP-hard combinatorial problem. In [25], CRO is employed to tackle this problem and is shown to outperform existing algorithms.

7 Concluding remarks and future work

CRO is a recently developed general-purpose optimization technique. Its inspiration comes from the nature of chemical reactions. It mimics the interactions of molecules, in the form of elementary reactions. The randomly constructed sequence of elementary reactions lets the molecules explore the solution space for the global minimum. Energy management is the fundamental characteristic of CRO. The conservation of energy governs the acceptance of new solutions and the scope of search. Its variable population structure allows the algorithm to adapt to the problem with reasonable mixture of intensification and diversification. These are the reasons why CRO performs very well in solving optimization problems. Although CRO is just recently proposed, it has been successfully applied to many benchmarks and practical problems. The examples described in this paper give readers some ideas on how to apply CRO to their own problems. We believe this is just the start of the CRO journey. This tutorial serves to summarize the current development of CRO and to lay down potential research directions.

Thanks to the No-Free-Lunch Theorem, each successful metaheuristic performs well on certain classes of problems. Which classes of problems are suitable for CRO? There is no easy answer at this moment. Similar to other evolutionary algorithms, when CRO is applied to more areas, the research community will give an answer. To ease implementation, a toolbox called CROToolbox is available.[8] Users can quickly employ CRO to their own problems and learn the characteristics of CRO with the toolbox. Moreover, there are very few efforts on parallelization and distributed computation of CRO. Due to its variable population structure, there is no strict requirement on the population size and synchronization among the distributed computational platforms. Furthermore, the frequencies of decomposition and synthesis affect the performance. How best to control these frequencies to further improve the performance is still an open question.

Acknowledgments This work was supported in part by the Strategic Research Theme of Information Technology of The University of Hong Kong. A.Y.S. Lam was also supported in part by the Croucher Foundation Research Fellowship.

References

1. AlRashidi M, El-Hawary M (2009) A survery of particle swarm optimization applications in electric power systems. IEEE Trans Evol Comput 13(4):913–918
2. Ashlock D (2004) Evolutionary computation for modeling and optimization. Springer, New York
3. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge, UK
4. Burger R (2000) The mathematical theory of selection, recombination, and mutation. Wiley, Chichester
5. Cela E (1998) The quadratic assignment problem: theory and algorithms. Kluwer Academic Publishers, Dordrecht, The Netherlands
6. Chen XS, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591–607
7. Demeulemeester EL, Herroelen WS (2002) Project scheduling: a research handbook. Academic Publishers, Boston, MA, USA
8. Dorigo M, Stutzle T (2004) Ant colony optimization. The MIT Press, Cambridge, MA, USA
9. Eiben AE (2001) Evolutionary algorithms and constraint satisfaction: definitions, survey, methodology, and research directions. Theoretical aspects of evolutionary computing. Springer, London, pp 13–30
10. Fortnow L (2009) The status of the P versus NP problem. Commun ACM 52(9):78–86
11. Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. WH Freeman & Co Ltd, New York
12. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
13. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA, USA
14. Guggenheim EA (1967) Thermodynamics: an advanced treatment for chemists and physicists. 5th edn. Wiley, North Holland
15. Kennedy J, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann, San Francisco
16. Kim M, Medard M, Aggarwal V, OReilly UM, Kim W, Ahn CW (2007) Evolutionary approaches to minimizing network coding resources. In: Proceedings of the 26th annual IEEE conference on computer Communications, Anchorage, AK, USA
17. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220((4598):671–680

[8] CROToolbox can be downloaded at http://cro.eee.hku.hk.

18. Lam AYS, Li VOK (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14(3):381–399
19. Lam AYS, Xu J, Li VOK (2010) Chemical reaction optimization for population transition in peer-to-peer live streaming. In: Proceedings of the IEEE congress on evolutionary computation. Barcelona, Spain
20. Lam AYS, Li VOK, Yu JJQ (2011, in press) Real-coded chemical reaction optimization. IEEE Trans Evol Comput (accepted for publication)
21. Lam AYS, Li VOK (2010) Chemical reaction optimization for cognitive radio spectrum allocation. In: Proceedings of the IEEE Global Communications Conference. Miami, FL, USA
22. Loiola EM, de Abreu NMM, Boaventura-Netto PO, Hahn P, Querido T (2007) A survey for the quadratic assignment problem. Eur J Oper Res 176(2):657–690
23. Ong YS, Lim MH, Chen XS (2010) Research frontier: memetic computation past, present and future. IEEE Comput Intell Mag 5(2):24–36
24. Palmes PP, Hayasaka T, Usui S (2005) Mutation-based genetic neural network. IEEE Trans Neural Netw 16(3):587–600
25. Pan B, Lam AYS, Li VOK (2011) Network coding optimization based on chemical reaction optimization. In: Proceedings of the IEEE global communications conference. Houston, TX, USA
26. Peng C, Zheng H, Zhao BY (2006) Utilization and fairness in spectrum assignment for opportunistic spectrum access. ACM/Kluwer Mobile Netw Appl 11(4):555–576
27. Ritchie G, Levine J (2004) A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. In: Proceedings of 23rd workshop of the UK planning and scheduling special interest group. Cork, Ireland
28. Price K, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
29. Rogers H (1987) Theory of recursive functions and effective computability. The MIT Press, Cambridge, MA, USA
30. Schach S (2010) Object-oriented and classical software engineering. 8th edn. McGraw-Hill, New York
31. Shadbolt N (2004) Nature-inspired computing. IEEE Intell Syst 19(1):2–3
32. Shin SY, Lee IH, Kim D, Zhang BT (2005) Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing. IEEE Trans Evol Comput 9(2):143–158
33. Subramanian AP, Gupta H, Das SR, Cao J (2008) Minimum interference channel assignment in multiradio wireless mesh networks. IEEE Trans Mobile Comput 7(12):1459–1473
34. Tollo GD, Roli A (2008) Metaheuristics for the portfolio selection problem. J Financial Quant Anal 8(4):621–636
35. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
36. Xu J, Lam AYS, Li VOK (2010) Chemical reaction optimization for the grid scheduling problem. In: Proceedings of the IEEE international conference on communications. Cape Town, South Africa
37. Xu J, Lam AYS, Li VOK (2011) Chemical reaction optimization for task scheduling in grid computing. IEEE Trans Parallel Distrib Syst 22(10):1624–1631
38. Xu J, Lam AYS, Li VOK (2011) Stock portfolio selection using chemical reaction optimization. In: Proceedings of the international conference on operations research and financial engineering. Paris, France
39. Xu J, Lam AYS, Li VOK (2010) Parallel chemical reaction optimization for the quadratic assignment problem. In: Proceedings of the international conference on genetic and evolutionary methods. Las Vegas, NV, USA
40. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
41. Yu JJQ, Lam AYS, Li VOK (2011) Evolutionary artificial neural network based on chemical reaction optimization. In: Proceedings of the IEEE congress on evolutionary computation. New Orleans, LA, USA
42. Yu L, Chen H, Wang S, Lai KK (2009) Evolving least squares support vector machines for stock market trend mining. IEEE Trans Evol Comput 13(1):87–102