Chemical Reaction Optimization for the Set Covering Problem

James J.Q. Yu, *Student Member, IEEE*
Department of Electrical and Electronic Engineering
The University of Hong Kong
Email: jqyu@eee.hku.hk

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

Victor O.K. Li, *Fellow, IEEE*
Department of Electrical and Electronic Engineering
The University of Hong Kong
Email: vli@eee.hku.hk

***Abstract*—The set covering problem (SCP) is one of the representative combinatorial optimization problems, having many practical applications. This paper investigates the development of an algorithm to solve SCP by employing chemical reaction optimization (CRO), a general-purpose metaheuristic. It is tested on a wide range of benchmark instances of SCP. The simulation results indicate that this algorithm gives outstanding performance compared with other heuristics and metaheuristics in solving SCP.**

***Index Terms*—Set covering problem, chemical reaction optimization, heuristic, metaheuristic.**

I. Introduction

The set covering problem (SCP) is one of the representative combinatorial optimization problems. It has many real-world applications, e.g. bus, railway and airline crew scheduling, vehicle routing, facility location, and political districting [1]. More recent applications of SCP are on sensor lifetime maximization [2] and phasor measurement unit placement [3].

SCP is formally defined as follows. We have a set of m elements $\mathbb{M} = \{1, \cdots, m\}$ and a collection of n subsets $\mathbb{N} = \{S_j \subseteq \mathbb{M}, 1 \leq j \leq n\}$, each of which is associated with a cost S_j, denoted as c_j. We say a collection of subsets $\mathbb{X} \subseteq \mathbb{N}$ is a cover of $\mathbb{M}$ if $\bigcup_{S_j \in \mathbb{X}} S_j = \mathbb{M}$ holds. $\mathbb{X}$ is a prime cover of $\mathbb{M}$ if there is no redundant subset in $\mathbb{X}$, i.e., $\mathbb{X}$ will not cover $\mathbb{M}$ if any subset is removed from $\mathbb{X}$. The goal of SCP is to find an $\mathbb{X}$ with the minimum cost.

SCP is usually formulated as a binary integer programming problem as follows:

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, \; i = 1, 2, \cdots, m, \\ & x_j \in \{0, 1\}, \; j = 1, 2, \cdots, n, \end{aligned} \tag{1}$$

where $a_{ij} = 1$ if $i \in S_j$ and $a_{ij} = 0$ otherwise. The decision variable x_j is set to one if subset S_j is selected in the cover $\mathbb{X}$.

It is also common to formulate SCP into matrix form. In this formulation, SCP is the problem of covering the rows of an $m \times n$ matrix by a subset of the columns at a minimal cost. We use $\mathbf{A} = \{a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n\}$ to represent the matrix, and we say the k-th element is covered by the l-th subset if $a_{kl} = 1$. We use $\mathsf{C} = \{c_j, 1 \leq j \leq n\}$ as the cost coefficient vector. Then SCP is defined as follows:

$$\begin{aligned} \min \quad & \mathsf{C}^\top \mathsf{X} \\ \text{s.t.} \quad & \mathbf{A}\mathsf{X} \geq \mathsf{b}, \end{aligned} \tag{2}$$

where $\mathsf{X} = \{x_j, 1 \leq j \leq n\}$ is the solution vector and b is the unit vector of length m. The constraint ensures that each row is covered by at least one column. If the costs for all subsets are identical, then the SCP is named *unicost* SCP. Otherwise, it is called *weighted* or *non-unicost* SCP.

SCP is known to be an NP-hard optimization problem [4], and metaheuristics have been shown to be effective in solving complex problems. Chemical reaction optimization (CRO) is one of the general-purpose metaheuristics which has shown its capability in solving similar NP-hard combinatorial problems, e.g. the quadratic assignment problem [5]. CRO is inspired by chemical reactions, where reactant molecules collide with container walls or with each other. During the collision, the structure of these molecules may change and the energy hold by these molecules may be transferred to other molecules or transformed into other energy forms. The changes made by the collision follow a natural tendency that the potential energy of product molecules is smaller than that of reactant molecules macroscopically [5]. CRO utilizes this tendency to perform optimization.

In this paper, we propose a heuristic-based CRO algorithm to solve SCP, named hCRO. We perform a series of simulations to test its performance. This paper is organized as follows. We perform a brief survey on previously proposed approaches to solve SCP, and some prior efforts on CRO in Section II. Section III presents the design and implementation of our proposed hCRO algorithm. The performance of our proposed algorithm is illustrated with the help of a series of benchmark problems in Section IV. Finally we conclude our work and propose some potential future work in Section V.

II. Background

SCP is a classical NP-hard combinatorial optimization problem, and has attracted the interests of researchers for several decades. Many exact algorithms, heuristic, and metaheuristic approaches have been proposed and reported in the literature.

Existing exact algorithms to solve SCP is mainly based on the branch-and-bound and branch-and-cut search algorithms [6]. Fisher and Kedia proposed an exact branch-and-bound algorithm based on a dual heuristic [7]. This algorithm is capable of solving SCP instances with up to 200 elements and 2000 subsets (200×2000). Beasley combined a Lagrangian-based heuristic, sub-gradient optimization, and linear programming to improve the branching strategy [8]. He then enhanced this algorithm using feasible solution exclusion constraints and Gomory's f-cut in [9]. The algorithm is tested on instances with matrices up to the order of 400×4000. Harche and Thompson developed a column subtraction exact algorithm to solve sparse instances of SCP up to the order of 900×8000 [10]. All the above exact algorithms are based on the tree-search algorithm, which has limitations such as extremely high computational complexity, very large searching space, and relatively poor performance [6]. Due to these drawbacks, researchers resorted to approximate algorithms to meet the requirement of less stringent computation with satisfactory solution quality.

Greedy algorithms are one of the heuristic approaches to quickly solve large combinatorial problems. Chvatal proposed the very first greedy algorithm to solve SCP in 1979 [11]. However, due to its deterministic and myopic feature, this algorithm can rarely generate good solutions despite it is fast and simple. In order to improve the performance of the canonical greedy algorithm, researchers have tried to introduce some stochastic features into it [12] [13]. Generally, greedy algorithms with stochastic features can generate better solutions than the canonical greedy ones. There are also some non-greedy-based heuristics. Caprara's work [14] is an example which gives good solutions. This work introduced variable fixing and pricing techniques into a Lagrangian heuristics.

The research on employing metaheuristic algorithms, especially evolutionary algorithms, to solve SCP has been intensely investigated in the last decade. A wide range of metaheuristics have been utilized, including genetic algorithm [15] [16], ant colony optimization [17], simulated annealing [18], and artificial neural networks [19]. The high adaptability and superior performance of metaheuristic make it a competitive approach to solve SCP. Moreover, due to the characteristics of unicost SCP, researchers have proposed some algorithms to solve this special kind of SCP. Grossman and Wool designed an artificial neural network framework to solve unicost SCP [20]. However, the existing algorithms to solve SCP have two drawbacks in general. Firstly, most algorithms are designed to solve non-unicost SCP. Beasley and Chu pointed out that their algorithms based on Lagrangian relaxation and genetic algorithm were not recommended for unicost problems [15]. From the literature, very few algorithms are found to work effectively for both unicost and non-unicost problem. Secondly, most algorithms which can generate satisfactory results are difficult to implement, while the simple algorithms, e.g., greedy algorithms, are less competitive in performance.

In order to overcome these two drawbacks, the goal of this work is to design a robust and simple metaheuristic based on CRO that can generate good results for both unicost and non-unicost SCP. CRO is a recently proposed general-purpose metaheuristic, which has been developed intensely in the past few years. CRO was proposed by Lam *et al.* in 2010 [5], and was originally designed for solving combinatorial optimization problems. They solved some classical problems, e.g., quadratic assignment problem and channel assignment problem. CRO is also applied to solve some real world applications like real-time monitoring [21] and smart grid [22]. For continuous problems, Lam *et al.* also proposed a variant of CRO, i.e., real-coded CRO, which has demonstrated superior performance in many real-world applications, e.g., training artificial neural networks [23] and optimal power flow problem in power grid [24].

III. Algorithm Design

In this section we will first introduce CRO. Then the implementation of operators used in CRO will be presented.

A. Chemical Reaction Optimization

We consider a number of molecules in a closed container with an attached energy buffer. Molecules are the basic operating agents of CRO. CRO manipulates and controls a collection of molecules to explore the solution space of an optimization problem. CRO considers the molecular structure as the feasible solution of the optimization problem. Besides the molecular structure, a molecule also has some other attributes that help the algorithm to perform optimization. Typically each molecule possesses two different kinds of energy, namely, potential energy (PE) and kinetic energy (KE). We use PE to represent the solution quality, or objective function value, of the corresponding molecular structure. The solution space is decribed by the potential energy surface (PES). The molecules can move freely on the PES. Every position on the PES is associated with a PE value. The lower the PE (for objective function minimization), the better is the solution. KE quantifies the ability of the molecule to move towards an area on the PES with higher values. The larger a molecule's KE, the higher it can position itself on the PES, which means that this molecule can accept worse solutions. This feature is very important for the cases when CRO tries to manipulate the molecules to jump out of a local optimum in the solution space.

CRO controls and manipulates the molecules with four different elementary reactions, namely, on-wall ineffective collision (*on-wall*), decomposition (*dec*), inter-molecular ineffective collision (*inter*), and synthesis (*syn*), each of which is described by an operator. Each operator modifies the molecular structures of some molecules, performing stochastic exploration or exploitation in the solution space. The four elementary reactions have different energy handling schemes and molecular structure operations. Meanwhile, they share a feature, which distinguishes CRO from other metaheuristics. All the operations in CRO must comply with the energy conservation law, which states that although energy is allowed to transform between types, the total energy in an isolated system (i.e. the container in CRO) shall remain constant. In CRO, the total energy of the system before and after an elementary reaction is constant. Interested reader can refer to [25] for the detailed implementation of CRO.

B. Encoding Scheme

There are two major encoding schemes to solve SCP in the literature. The first one is the natural encoding scheme. This scheme uses a binary vector of length n for a solution. Each element in the vector represents the status of one particular subset. In other words, this encoding scheme uses the X in (2) as the solution for optimization. This encoding scheme is easy to implement, but a random solution generated from this scheme is not guaranteed to be a cover of $\mathbb{M}$. In order to overcome this drawback, a second type of encoding scheme is developed for SCP.

The second encoding scheme, which we adopt to solve SCP with CRO, has each value in the solution vector representing a subset index. The values are selected from the indices of those subsets that cover the corresponding element. For example, consider a solution vector $[2, 6, 7, 2, \cdots]$. This solution uses the second subset to cover the first and fourth elements in $\mathbb{M}$, and the sixth subset to cover the second element and so forth. Solutions in the second scheme have shorter length than those in the first in general (for most non-unicost SCP, m is smaller than n). Moreover, all the solutions generated by this encoding scheme satisfy the constraints of SCP naturally [6].

C. Algorithm Design

CRO defines four different types of elementary reactions, which possess different functionalities. So we design a corresponding operator for each of them. We also design an initial solution generator to generate the solution structures of new molecules. We generate all random numbers uniformly in the solution space, unless stated otherwise.

1) Initial Solution Generator: This operator is applied whenever a new molecule is generated. Instead of randomly assigning subset indices (this is common when we solve other optimization problems), we use a reverse cumulative scheme to select a random subset to cover this element. The scheme is stated in Algorithm 1.

In Algorithm 1, the variables are calculated by

$$v_j = c_{max} + c_{min} - c_j \tag{3}$$

and

$$p_j = \frac{v_j}{\sum_{k=1}^{|\mathbb{X}_i|} v_k} \tag{4}$$

This scheme renders the initial solution more likely to include those subsets with lower cost, while ensuring the solution complies with all constraints. This scheme assigns the subsets with lower cost a larger possibility of being selected

Algorithm 1 REVERSE CUMULATIVE SCHEME

```
1: for all Elements $i$ in a solution do
2:     Find all subsets $\mathbb{X}_i$ that cover the i-th element in $\mathbb{M}$.
3:     Find the maximum cost $c_{max}$ and the minimum cost
         $c_{min}$ of subsets in $\mathbb{X}_i$.
4:     for all $S_j \subseteq \mathbb{X}_i$ do
5:         Assign a reverse cumulative value $v_j$.
6:         Assign its probability of being selected $p_j$.
7:     end for
8:     Select the value of Element $i$ according to each
         subsets' $p_j$.
9: end for
```

by (3) and (4). For example, the first element in $\mathbb{M}$ can be covered by Subsets 1, 3, and 6, whose costs are 2, 4, and 5, respectively. Then the reverse cumulative value of Subset 1 is calculated by $5 + 2 - 2 = 5$, and that for Subsets 3 and 6 are 3 and 2. Thus the probability of being selected are 0.5, 0.3, and 0.2, respectively.

2) Neighborhood Search Operator: This operator is applied to the two ineffective elementary reactions, namely, on-wall and inter. This operator modifies the input solution slightly to perform a local search. We employ a perturbation heuristic to act as the neighborhood search operator in CRO. The algorithm is stated in Algorithm 2.

Algorithm 2 PERTURBATION HEURISTIC

```
1: for all Subsets $S_j$ in the solution do
2:     Calculate its cost efficiency value $e_j^c$.
3: end for
4: Find the Subset $S_i$ with the lowest cost efficiency.
5: Remove all $i$ in the solution and leave blanks.
6: while There are blanks in the solution do
7:     for all Subsets $S_k \subseteq \mathbb{N}$ do
8:         Calculate its repair efficiency value $e_k^r$
9:     end for
10:    for all Subsets $S_k \subseteq \mathbb{N}$ do
11:        Calculate its probability of being selected $p_k$
12:    end for
13:    Select a subset $S_l$to repair the solution according to
         each subsets' $p_k$.
14:    Use $l$ to fill all the blanks which can be covered by
         $S_l$.
15: end while
```

In Algorithm 2, the variables are calculated by

$$e_j^c = \frac{n_j}{c_j}, \tag{5}$$

where n_j is the occurrence of j in the solution, and

$$e_k^r = \frac{s_k}{c_k}, \tag{6}$$

where s_k is the number of blanks that Subset S_k can cover in the solution, and

$$p_k = \frac{e_k^r}{\sum_{j=1}^{|\mathbb{X}|} e_j^r} \tag{7}$$

TABLE I
BEASLEY'S OR LIBRARY NON-UNICOST INSTANCES

Set	Instances	Size ($m \times n$)	Costs	Density	Opt. Solution
4	10	200×1000	1-100	2%	Known
5	10	200×2000	1-100	2%	Known
6	5	200×1000	1-100	5%	Known
A	5	300×3000	1-100	2%	Known
B	5	300×3000	1-100	5%	Known
C	5	400×4000	1-100	2%	Known
D	5	400×4000	1-100	5%	Known
NRE	5	500×5000	1-100	10%	Unknown
NRF	5	500×5000	1-100	20%	Unknown
NRG	5	1000×10000	1-100	2%	Unknown
NRH	5	1000×10000	1-100	5%	Unknown

TABLE II
BEASLEY'S OR LIBRARY UNICOST INSTANCES

Set	Instances	Size ($m \times n$)	Density	Opt. Solution
E	5	50×500	2%	Known
CLR.10	1	511×210	2%	Unknown
CLR.11	1	1023×330	5%	Unknown
CLR.12	1	2047×495	2%	Unknown
CLR.13	1	4095×715	5%	Unknown
CYC.6	1	240×192	2%	Known
CYC.7	1	672×448	5%	Unknown
CYC.8	1	1792×1024	10%	Unknown
CYC.9	1	4608×2304	20%	Unknown
CYC.10	1	11520×5120	2%	Unknown
CYC.11	1	28160×11264	5%	Unknown

This perturbation heuristic can be divided into two major parts: remove and repair. In the remove phase (Lines 1 to 5 of Algorithm 2), the least efficient subset is removed from the solution. Its cost efficiency value is calculated by (5). Assume a Subset 2 with cost 10. If it covers two elements in the solution, its cost efficiency value is 5. At the end of the remove phase, we have an incomplete solution with one or several blank positions. Then in the repair phase, we select a most efficient subset to fill in the blank(s). This repair efficiency value is calculated by (6). With the repair efficiency values, we can calculate the probability of being selected for repairing using (7). For example, assume we have an incomplete solution $[1, \_, 4, \_, \_, \cdots]$ where the underline positions are blank positions. Subset 3 can cover two blanks with a cost 20, and Subset 5 can cover three blanks with a cost 40. So the repair efficiency values of Subsets 3 and 5 are 0.1 and 0.075, respectively. If there are no other subsets, the probabilities of selecting Subsets 3 and 5 are 57% and 43%, respectively. If the algorithm chooses Subset 3 for repairing, the two blanks which can be covered by Subset 3 is filled with 3. This completes one iteration of the repairing phase and this process iterates until all blanks are filled.

For on-wall, we directly employ this perturbation scheme to the molecule, while for inter, we apply the scheme to the two input molecules simultaneously.

3) Dec and Syn Operators: The main purpose of dec and syn is to help the molecules jump out of local optimum. So we usually impose severe changes to the input molecule(s). For dec operator, we first copy the input molecule to the two output molecules, then perform the neighborhood search operator on each molecule for 10 times separately. The resultant molecules are regarded as the final output molecules. For the syn operator, we introduce a probabilistic combination scheme to combine the two input molecules and create a new one. Assume the two input solutions are X_1 and X_2, and their costs are c_1 and c_2, respectively. Each element in the output new solution is drawn from the same position of either X_1 or X_2 with the probability of $\frac{c_2}{c_1+c_2}$ and $\frac{c_1}{c_1+c_2}$, respectively.

IV. SIMULATION RESULTS

In this section we will first introduce the benchmark instances used to evaluate the performance of our proposed hCRO algorithm. Then the detailed simulation parameter settings, results, and comparisons are presented.

TABLE III
hCRO PARAMETER VALUES

Parameter	Value
Initial population size	10
Initial molecular kinetic energy	1000
Initial central energy buffer size	10000
Collision rate	0.1
Energy loss rate	0.1
Decomposition threshold	10000
Synthesis threshold	1000

A. Benchmark Instances

We will test the performance of our proposed hCRO using 65 non-unicost SCP test instances from Beasley's OR Library [26]. We note that almost all SCP algorithms developed in the past two decades were tested using these problems. The instances are divided into 11 different sets, as listed in Table I, where the density is the percentage of non-zero entries in the SCP matrix $\mathbf{A}$.

We also test the performance of hCRO with the unicost instances in Beasley's OR Library and the information about these instances are listed in Table II.

B. Parameter Tuning and Simulation Environment

When applying hCRO to perform simulation, several parameters must be set. Coy *et al.* note that it is often very difficult to find appropriate parameter settings for metaheuristics, and common procedures of generating proper parameter values have ranged from simple trial-and-error to complicated sensitivity analysis [27]. In this work, we use the trial-and-error method to tune the hCRO parameters, as in some previous CRO efforts [21] [28]. The first instances in the SCP problem set 4, 5, 6, A, B, C, and D are selected as representative instance for parameter tuning and we perform a series of test runs on these instances, using the methodology described in [28]. The final parameters used for all test instances are listed in Table III.

Our proposed approach was implemented in C++ on a computer with an Intel Core i5 3.1-GHz processor and MinGW compiler. In our experimental study, 100 trials of hCRO were conducted for each of these test problems. The maximum function evaluation limit is set to $n \times 1000$, which is smaller or equal to that in all the metaheuristics we compare with.

C. Simulation Results and Comparison with Other Algorithms

The simulation results are presented in Table IV. In this table, "Inst." is the instance index, "BKS" is the optimum solution or the best known value, "Opt." is the number of trials that hCRO is able to find the optimum solution or the best known value. We also present the mean, best, and worst objective function values of the 100 trials, and their respective percentages above the optimal value. The results of hCRO on unicost SCP instances are also presented in Table V.

From Tables IV and V, we can see that hCRO has excellent performance in non-unicost instances, and all 65 optimum solutions are generated in every run for all instances. For unicost-SCP instances, hCRO obtains 12 optimums out of 15. For the remaining instances, hCRO can also generate a satisfactory result (error percentage around 1% of best results).

In order to further demonstrate the performance of hCRO, we also compare hCRO with other algorithms proposed in the recent literature. We compare the performance on non-unicost instances of hCRO with the Lagrangian heuristic by Beasley (Be) [26], the genetic algorithm by Beasley and Chu (BeCh) [15], the Lagrangian heuristic by Caprara *et al.* (CFT) [14], a probabilistic greedy search heuristic by Haouari and Chaouachi (PROG) [13], an indirect genetic algorithm by Aickelin (IGA) [16], a metaheuristic for randomized priority search by Lan *et al.* (RaPS) [29], and a metaheuristic algorithm based on gravity by Balachandar and Kannan (GRA) [6]. The comparison is presented in Table VI, where the row of "Opt. Found" is the number of instances in which the corresponding algorithm finds the optimal value or the best known solution out of the all 65 instances. The table shows the average gap of the global optimal found by the algorithm and BKS. For example, an algorithm generates an optimal value of 52 on a problem instance with a best known value of 50, the gap is defined as $(52 - 50)/50 = 4\%$. From the table we can see, hCRO is one of the best four algorithms that can find the optimal value or best known solution 100% of the time.

We further demonstrate the performance of hCRO by comparing the simulation results on unicost instances with other algorithms. These algorithms include the heuristic random approximation by Peleg *et al.* (RR) [30], a greedy heuristic by Chvatal (Gr) [11], three algorithms proposed by Grossman and Wool in [20] (Alt-Gr, NN, and R-Gr), and a mean-field approach by Ohlsson *et al.* (MF) [19]. The results are presented in Table VII. From the comparison we can see hCRO again outperforms all other algorithms dramatically. Therefore, hCRO is an effective algorithm in solving both non-unicost and unicost set covering problems.

D. Analysis on Performance Contribution of the Proposed Heuristic Schemes and CRO Framework

In order to analyze the contribution of different proposed heuristic schemes and the CRO framework to the outstanding

TABLE IV
hCRO RESULTS ON NON-UNICOST INSTANCES

Inst.	BKS	Opt.	Best		Mean		Worst	
			Value	Pct.	Value	Pct.	Value	Pct.
4.1	429	100	429	0	429	0	429	0
4.2	512	100	512	0	512	0	512	0
4.3	516	100	516	0	516	0	516	0
4.4	494	100	494	0	494	0	494	0
4.5	512	100	512	0	512	0	512	0
4.6	560	100	560	0	560	0	560	0
4.7	430	100	430	0	430	0	430	0
4.8	492	100	492	0	492	0	492	0
4.9	641	100	641	0	641	0	641	0
4.10	514	100	514	0	514	0	514	0
5.1	253	100	253	0	253	0	253	0
5.2	302	100	302	0	302	0	302	0
5.3	226	100	226	0	226	0	226	0
5.4	242	100	242	0	242	0	242	0
5.5	211	100	211	0	211	0	211	0
5.6	213	100	213	0	213	0	213	0
5.7	293	100	293	0	293	0	293	0
5.8	288	100	288	0	288	0	288	0
5.9	279	100	279	0	279	0	279	0
5.10	265	100	265	0	265	0	265	0
6.1	138	100	138	0	138	0	138	0
6.2	146	100	146	0	146	0	146	0
6.3	145	100	145	0	145	0	145	0
6.4	131	100	131	0	131	0	131	0
6.5	161	100	161	0	161	0	161	0
A.1	253	100	253	0	253	0	253	0
A.2	252	100	252	0	252	0	252	0
A.3	232	100	232	0	232	0	232	0
A.4	234	100	234	0	234	0	234	0
A.5	236	100	236	0	236	0	236	0
B.1	69	100	69	0	69	0	69	0
B.2	76	100	76	0	76	0	76	0
B.3	80	100	80	0	80	0	80	0
B.4	79	100	79	0	79	0	79	0
B.5	72	100	72	0	72	0	72	0
C.1	227	100	227	0	227	0	227	0
C.2	219	100	219	0	219	0	219	0
C.3	243	100	243	0	243	0	243	0
C.4	219	100	219	0	219	0	219	0
C.5	215	100	215	0	215	0	215	0
D.1	60	100	60	0	60	0	60	0
D.2	66	100	66	0	66	0	66	0
D.3	72	100	72	0	72	0	72	0
D.4	62	100	62	0	62	0	62	0
D.5	61	100	61	0	61	0	61	0
NRE.1	29	100	29	0	29	0	29	0
NRE.2	30	100	30	0	30	0	30	0
NRE.3	27	100	27	0	27	0	27	0
NRE.4	28	100	28	0	28	0	28	0
NRE.5	28	100	28	0	28	0	28	0
NRF.1	14	100	14	0	14	0	14	0
NRF.2	15	100	15	0	15	0	15	0
NRF.3	14	100	14	0	14	0	14	0
NRF.4	14	100	14	0	14	0	14	0
NRF.5	13	100	13	0	13	0	13	0
NRG.1	176	100	176	0	176	0	176	0
NRG.2	154	100	154	0	154	0	154	0
NRG.3	166	100	166	0	166	0	166	0
NRG.4	168	100	168	0	168	0	168	0
NRG.5	168	100	168	0	168	0	168	0
NRH.1	63	100	63	0	63	0	63	0
NRH.2	63	100	63	0	63	0	63	0
NRH.3	59	100	59	0	59	0	59	0
NRH.4	58	100	58	0	58	0	58	0
NRH.5	55	100	55	0	55	0	55	0

TABLE V
hCRO Results on Non-unicost Instances

Inst.	BKS	Opt.	Best		Mean		Worst	
			Value	Pct.	Value	Pct.	Value	Pct.
E.1	5	100	5	0	5	0	5	0
E.2	5	100	5	0	5	0	5	0
E.3	5	100	5	0	5	0	5	0
E.4	5	100	5	0	5	0	5	0
E.5	5	100	5	0	5	0	5	0
CLR.10	25	100	25	0	25	0	25	0
CLR.11	23	100	23	0	23	0	23	0
CLR.12	23	100	23	0	23	0	23	0
CLR.13	23	100	23	0	23	0	23	0
CYC.6	60	100	60	0	60	0	60	0
CYC.7	144	11	144	0	146.7	0.0188	152	0.0556
CYC.8	344	14	344	0	349.2	0.0151	361	0.0494
CYC.9	780	0	789	0.0115	797.3	0.0221	819	0.05
CYC.10	1792	0	1802	0.0056	1832.6	0.0226	1872	0.0446
CYC.11	4103	0	4113	0.0024	4159.2	0.0137	4201	0.0239

TABLE VI
Performance Gap Comparison on Non-unicost Instances (%)

Problem Set	hCRO	RaPS	GRA	CFT	BeCh	IGA	PROG	Be	Greedy
4	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.06	3.78
5	0.00	0.00	0.00	0.00	0.09	0.00	0.88	0.18	5.51
6	0.00	0.00	0.00	0.00	0.00	0.00	0.69	0.56	7.22
A	0.00	0.00	0.00	0.00	0.00	0.00	0.75	0.82	5.61
B	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.81	5.57
C	0.00	0.00	0.00	0.00	0.00	0.00	0.87	1.93	6.88
D	0.00	0.00	0.00	0.00	0.00	0.32	0.00	2.75	9.79
NRE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.50	12.75
NRF	0.00	0.00	0.00	0.00	0.00	0.00	1.43	7.16	12.98
NRG	0.00	0.00	0.00	0.00	0.13	0.13	1.18	4.83	8.49
NRH	0.00	0.00	0.00	0.00	0.63	1.30	1.68	8.12	11.78
Overall	0.00	0.00	0.00	0.00	0.08	0.16	0.72	2.36	8.21
Opt. Found	65/65	65/65	65/65	65/65	61/65	61/65	22/65	20/65	0/65

TABLE VII
Performance Gap Comparison on Unicost Instances (%)

Problem Set	hCRO	RR	Gr	Alt-Gr	NN	R-Gr	MF
Overall	0.62	20.34	10.50	7.29	10.07	9.85	6.51

performance of hCRO, we also construct different hCRO variations and a heuristic-based Genetic Algorithm (hGA).

To determine the impact of our initialization operator (Algorithm 1) and our neighborhood search operator (Algorithm 2), we create alternative operators for these two schemes and construct two new hCRO variations:

- hCRO/IR: This variation of hCRO utilizes our previously proposed perturbation heuristic as the neighborhood search operator, with a "random pick scheme" as the initial population generator. In the "random pick scheme", we first construct a permutation vector of all elements in $\mathbb{M}$. We then start from the first element i in this permutation vector and find all subsets $\mathbb{X}_i$ that cover i. Then we randomly select one such subset and go on to the next element j. If j is not covered yet, we repeat the previous step, i.e. find all subsets that cover j and randomly select one. If j is already covered, we go to the next element. This process repeats until all elements are covered by at least one subset.
- hCRO/NR: This variation of hCRO utilizes our previously proposed reverse cumulative scheme as the initial population generator, with a "remove-repair scheme" as the neighborhood search operator. In the "remove-repair scheme", we first randomly remove one subset in the solution. Then we determine all elements not yet covered by any subsets. The remaining part is similar to the "random pick scheme", where we build a permutation vector, find all subsets that cover the current element and randomly select one such subset.

Besides hCRO/IR and hCRO/NR, we also develop hGA to solve SCP in order to reveal the contribution of the CRO framework in hCRO:

- hGA: We utilize our previously proposed reverse cumulative scheme as the initial population generator. We use the inter-molecular ineffective collision operator in hCRO as the crossover operator in hGA, i.e., two perturbation heuristics are executed simultaneously. As for the mutation operator, we adopt the decomposition operator in

TABLE VIII
PERFORMANCE GAP COMPARISON ON hCRO, hCRO/IR, hCRO/NR, AND hGA (%)

Problem Set	hCRO	hCRO/IR	hCRO/NR	hGA
Non-unicost	0.00	0.04	1.78	0.11
Unicost	0.62	2.19	14.64	3.82

hCRO and select the better-performing solution as the mutated chromosome. We set the population size the same with hCRO. The crossover rate and mutation rate are set at 0.8 and 0.2, respectively, which is a commonly used combination of parameters for GA.

We perform simulations of hCRO, hCRO/IR, hCRO/NR, and hGA on both unicost and non-unicost instances. The simulation results are presented in Table VIII.

From the results we can see that hCRO always performs the best. While hCRO/IR can generate similar results as hCRO, there is still a small gap between the performance of the two algorithms. This demonstrates the superiority of the reverse cumulative scheme. hCRO/NR performs much worse than all other algorithms, and this observation underlines the importance of the perturbation heuristic in generating good results. The results also show that the CRO framework is superior to conventional problem solver frameworks like GA. This is probably due to the unique energy conservation design in CRO as well as its tolerant, energy-related individual selection pattern [5].

V. CONCLUSION

In this paper we develop a heuristic-based CRO algorithm to solve non-unicost and unicost SCP. This algorithm introduces two heuristics into the operators of CRO. We study the performance of hCRO with a series of benchmark test instances from the Beasley's OR Library and show that hCRO enjoys superior performance in terms of the solution quality when compared with other algorithms. hCRO is able to find all 65 optimal solutions in non-unicost instances and it demonstrates outstanding performance when applied to unicost SCP. We also perform a series of test over different variations of hCRO as well as a heuristic-based GA to demonstrate the contribution of these two heuristics and the CRO framework on the final performance.

Further research includes the improvement of the performance for huge SCP problems. We can also introduce some implementation techniques that are commonly used by other SCP algorithms, e.g., group memory between iterations, to CRO. Some of the heuristics developed in our proposed algorithm can also be applied to solve other applications of CRO, such as the bin packing problem and the multi-dimensional knapsack problem. Last but not least, we can use the proposed algorithm to solve real-world applications of SCP.

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

[1] E. Balas and M. W. Padberg, "Set partitioning: A survey," *SIAM Review*, vol. 18, no. 4, pp. 710–761, 1976.

[2] X.-M. Hu, J. Zhang, Y. Yu, H.-H. Chung, Y.-L. Li, Y.-H. Shi, and X.-N. Luo, "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 766–781, 2010.

[3] J. Xu, M. H. F. Wen, K.-C. Leung, and V. O. K. Li, "Optimal PMU placement using chemical reaction optimization," in *Proc. IEEE Innovative Smart Grid Technologies Conference (ISGT)*, Washington DC, U.S., Feb. 2013, pp. 1–6.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

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

[6] S. R. Balachandar and K. Kannan, "A meta-heuristic algorithm for set covering problem based on gravity," *International Journal of Computational and Mathematical Sciences*, vol. 4, no. 5, pp. 223–228, 2010.

[7] M. L. Fisher and P. Kedia, "Optimal solution of set covering/partitioning problems using dual heuristics," *Management Science*, vol. 36, no. 6, pp. 674–688, 1990.

[8] J. E. Beasley, "An algorithm for set covering problem," *Eur. J. Oper. Res.*, vol. 31, no. 1, pp. 85–93, 1987.

[9] J. E. Beasley and K. Jornsten, "Enhancing an algorithm for set covering problems," *Eur. J. Oper. Res.*, vol. 58, no. 2, pp. 293–300, 1992.

[10] F. Harche and G. L. Thompson, "The column subtraction algorithm: An exact method for solving weighted set covering, packing and partitioning problems," *Computers and Operations Research*, vol. 21, no. 6, pp. 689–705, 1994.

[11] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[12] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[13] M. Haouari and J. S. Chaouachi, "A probabilistic greedy search algorithm for combinatorial optimization with application to the set covering problem," *Journal of the Operational Research Society*, vol. 53, pp. 792–799, 2002.

[14] A. Caprara, M. Fischetti, and P. Toth, "A heuristic method for the set covering problem," *Operations Research*, vol. 47, pp. 730–743, 1999.

[15] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *Eur. J. Oper. Res.*, vol. 94, no. 2, pp. 392–404, 1996.

[16] U. Aickelin, "An indirect genetic algorithm for set covering problems," *Journal of the Operational Research Society*, vol. 53, no. 10, pp. 1118–1126, 2002.

[17] L. Lessing, I. Dumitrescu, and T. Stutzle, "A comparison between ACO algorithms for the set covering problem," *Ant Colony Optimization and Swarm Intelligence*, vol. 3172, pp. 1–12, 2004.

[18] L. W. Jacobs and M. J. Brusco, "Note: A local-search heuristic for large set-covering problems," *Naval Research Logistics*, vol. 42, no. 7, pp. 1129C–1140, 1995.

[19] M. Ohlsson, C. Peterson, and B. Soderberg, "An efficient mean field approach to the set covering problem," *Eur. J. Oper. Res.*, vol. 133, no. 3, pp. 583C–595, 2001.

[20] T. Grossman and A. Wool, "Computational experience with approximation algorithms for the set covering problem," *Eur. J. Oper. Res.*, vol. 101, no. 1, pp. 81–92, 1997.

[21] J. J. Q. Yu, V. O. K. Li, and A. Y. S. Lam, "Sensor deployment for air pollution monitoring using public transportation system," in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Brisbane, Australia, Jun. 2012, pp. 1–7.

[22] ——, "Optimal V2G scheduling of electric vehicles and unit commitment using chemical reaction optimization," in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Cancun, Mexico, Jun. 2013, pp. 392–399.

[23] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, “Evolutionary artificial neural network based on chemical reaction optimization,” in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, LA, U.S., Jun. 2011, pp. 2083–2090.

[24] Y. Sun, A. Y. S. Lam, V. O. K. Li, J. Xu, and J. J. Q. Yu, “Chemical reaction optimization for the optimal power flow problem,” in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Brisbane, Australia, Jun. 2012, pp. 1–8.

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

[26] J. E. Beasley, “A lagrangian heuristic for set-covering problems,” *Naval Research Logistics*, vol. 37, no. 1, pp. 151–164, 1990.

[27] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil, “Using experimental design to find effective parameter settings for heuristics,” *Journal of Heuristics*, vol. 7, pp. 77–97, 2001.

[28] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, “Real-coded chemical reaction optimization with different perturbation functions,” in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Brisbane, Australia, Jun. 2012, pp. 1–8.

[29] G. Lan, G. W. DePuy, and G. E. Whitehouse, “An effective and simple heuristic for the set covering problem,” *Eur. J. Oper. Res.*, vol. 176, pp. 1387–1403, 2007.

[30] D. Peleg, G. Schechtman, and A. Wool, “Randomized approximation of bounded multicovering problems,” *Algorithmica*, vol. 18, no. 1, pp. 44–66, 1997.