Soft Comput (2017) 21:6421–6433
DOI 10.1007/s00500-016-2200-4

METHODOLOGIES AND APPLICATION

Chemical reaction optimization with unified tabu search for the vehicle routing problem

Thu-Lan Dam[1,2] · **Kenli Li**[1,3,4] · **Philippe Fournier-Viger**[5]

Published online: 27 May 2016

Abstract This study proposes a new approach combining the chemical reaction optimization framework with the unified tabu search (UTS) heuristic to solve the capacitated vehicle routing problem (CVRP). The CVRP is one of the most well-studied problems not only because of its real-life applications but also due to the fact that the CVRP could be used to evaluate the efficiency of new algorithms and optimization methods. Chemical reaction optimization (CRO) is a new optimization framework mimicking the nature of chemical reactions. The CRO method has proved to be very effective for solving NP-hard optimization problems such as the quadratic assignment problem, neural network training, the knapsack problem, and the traveling salesman problem. We also present the design of elementary chemical reaction operations, the adaptation of the UTS algorithm to educate solutions in these operations. Finally, a thorough testing against well-known benchmark problems has been conducted. Experimental results show that the proposed algorithm is efficient and highly competitive in comparison with several prominent algorithms for this problem. The presented methodology may be a fine approach for developing similar algorithms to address other routing variants.

Keywords Capacitated vehicle routing problem · Chemical reaction optimization · Tabu search · Metaheuristic

Communicated by V. Loia.

✉ Kenli Li
lkl@hnu.edu.cn

Thu-Lan Dam
lanfict@gmail.com

Philippe Fournier-Viger
philfv@hitsz.edu.cn

1 College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

2 Faculty of Information Technology, Hanoi University of Industry, Hanoi, Vietnam

3 CIC of HPC, National University of Defense Technology, Changsha 410073, China

4 National Supercomputing Center in Changsha, Changsha 410082, China

5 School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, Guangdong, China

1 Introduction

The vehicle routing problem (VRP) plays an important role in the optimization of transportation cost for logistics and supply chain management. The VRP is also one of the most extensively studied problems due to its methodological interest. The VRP or the CVRP was first introduced by Dantzig and Ramser (1959). It is also a basic model for a large number of routing problems. In the VRP literature, there are many contributions targeted at this basic problem. The CVRP is described as the problem in which vehicles concentrated on a single depot are required to visit a number of customers in order to service their known demands. The total transportation cost of vehicles must be minimized. The CVRP is an NP-hard problem. Hence, it is difficult to solve the CVRP using exact algorithms in reasonable computing time. Therefore, heuristic and metaheuristic methods have mostly been used to address this problem; for example, see a number of recent surveys (Laporte et al. 2000; Cordeau et al. 2002; Eksioglu et al. 2009) and books (Toth and Vigo 2002; Golden et al. 2008).

Chemical reaction optimization is a fairly new paradigm introduced recently by Lam and Li (2010). This paradigm

encodes solutions as molecules and mimics the interactions of molecules in chemical reactions to search for optimal solutions. The CRO has four elementary reactions, namely, on-wall ineffective collision, inter-molecular ineffective collision, decomposition and synthesis. The first two reactions perform the exploitation, while the last two reactions gain the exploration. Meanwhile, a central energy buffer is also implemented in CRO and helps the molecules to escape from local optima. The CRO method has a good searching ability. It can also inherit advantages of genetic algorithms (GAs) when the four operators of CRO are designed using the crossover and mutation operators. The CRO method has demonstrated its capability to solve NP-hard optimization problems. Specifically, this paradigm has been applied to address many problems with high efficiency as presented in (Lam and Li 2012; Li et al. 2012; Xu et al. 2013; Truong et al. 2013; Melin et al. 2013; Sánchez et al. 2014; Truong et al. 2015).

In order to evaluate the effectiveness of algorithms for solving the CVRP, the most used benchmark is the 14 classical instances of Christofides et al. (1979). Besides, other well-known test instance sets for the CVRP have been provided by authors like Augerat et al., Breedam, Christofides, and Elion. Each instance has its particular characteristics such as the number of customers, constraints on vehicle capacity, restrictions on maximum route lengths, and a predefined minimal number of vehicles. Many algorithms have provided good solutions for the CVRP but they may not pay much attention to constraints that exist in real life. For example, most works focus on minimizing the total travel cost than the number of vehicles used. Unified tabu search (UTS) (Cordeau et al. 2001) is an effective and flexible local search metaheuristic which relaxes the constraints and handles them through a penalty function when exploring neighborhoods. However, the UTS performs the single-starting-point search and thus its performance relies highly on a good initial solution. Recently, the optimized crossover genetic algorithm of Nazif and Lee (2012) was shown to produce very good solutions, though the computation time still remains high. Moreover, novel optimization paradigms should be able to perform well in comparison with other optimization techniques and must be flexible to handle different variants of the VRP (Laporte 2009). Furthermore, the CRO method has been shown to be a successful optimization algorithm for NP-hard problems but it should be improved in many aspects, especially its exploitation capability. And the CRO has not yet been applied to solve the CVRP. Hence, it is worthwhile to evaluate the performance of the CRO paradigm for addressing the CVRP.

Given the above observations, we propose an algorithm that combines the CRO framework and the adapted UTS algorithm to solve the CVRP. The main contribution of this work is to develop a new approach to tackle the CVRP by adapting the conventional CRO framework in combination with the adapted UTS, and to design new operations for performing the four elementary chemical reactions. The inter-molecular ineffective collision operator is a specialized crossover which aims at minimizing the number of vehicles used and the total travel cost concurrently while satisfying constraints. The remaining reaction operators are well-designed by adopting crossover and mutation operators of GAs. Offspring produced by these crossovers and mutation are likely to violate some of the constraints, which is an important issue for GAs. Therefore, the adapted UTS procedure is employed to educate the resulting solutions produced by crossover or mutation operator in each CRO reaction. The adapted UTS is able to repair solutions and enhance their quality. Penalty parameters of the adapted UTS are updated based on information of the complete population instead of using only the current solution as in the original UTS. The adapted UTS not only helps the algorithm to get fast convergence but also enriches the diversification, which could improve exploitation capability of the canonical CRO. By this design, the proposed method, called CROUTS, inherits the advantages of both GA and UTS. The total distance travelled by the vehicles and the number of vehicles are minimized at the same time while relaxing constraints. Furthermore, the decomposition and synthesis reactions may change the number of individuals in a population. Hence the proposed algorithm has more opportunity to jump out of local optima as well as to explore broader areas of the solution space. The CROUTS can thus find good solutions in less time than a GA. On the other hand, the proposed method suffers from the inherent drawback of the CRO paradigm, which is that the number of parameters is slightly greater than some other approaches. However, these parameter values are deduced from the literature and well-tuned in the extensive experiment contributing to the effectiveness of the proposed algorithm, as well as guiding other researchers for future work. Thirty benchmark instances of the CVRP are employed to confirm the efficiency of the proposed algorithm. Experimental results show that the proposed algorithm is efficient and highly competitive in comparison with other algorithms from the literature. The proposed algorithm is able to find high-quality solutions within a reasonable time.

The rest of this paper is organized as follows: The next section describes the vehicle routing problem then briefly presents the literature review. Related work involving the chemical reaction algorithm, the unified tabu search and the genetic algorithm are briefly presented in Sect. 3. In Sect. 4, the proposed CROUTS algorithm is introduced and analyzed in detail. Then, Sect. 5 surveys and compares the performance of the proposed CROUTS algorithm. Finally, the last section draws the conclusion and discusses future work.

2 The vehicle routing problem

The vehicle routing problem is formally defined on a complete undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_0, v_1, \ldots, v_n\}$ is the vertex set and $\mathcal{E} = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$ is the edge set. Vertex v_0 refers to the depot while the remaining vertices $v_i = (v_1, v_2, \ldots, v_n)$ are customers. A distance or travel cost c_{ij} is assigned to every edge $(v_i, v_j) \in \mathcal{E}$. Each customer v_i has a demand q_i and a service time d_i. We have a fleet of m vehicles, each vehicle has capacity Q_k and is based at the depot. A route is a sequence of customers that the vehicle services. Vehicle routes are restricted to a maximum duration of $D_k, k = 1, \ldots, m$.

In this study, we address the case where vehicles are homogenous with $Q_k = Q$ and have a common route duration restriction $D_k = D$. Therefore, the objective of the CVRP is to construct a feasible set of at most m routes with one route for each vehicle, minimizing the total cost, such as (1) the vehicle must start and finish its tour at the depot; (2) each customer must be serviced by exactly one vehicle; (3) the total size of demand must not exceed Q and vehicle route duration combining travel and service time is bounded to a preset limit duration D. Mathematical formulations of the CVRP can be found in the book of Toth and Vigo (2002).

The CVRP is one of the most popular problems in combinatorial optimization. Hence, the CVRP literature is very rich. As the CVRP is an NP-hard problem, a large number of approximation techniques have been proposed to solve this one. These techniques are divided into two main categories: classical heuristics and metaheuristics. An overview of these techniques may also be found in (Laporte et al. 2000) and (Cordeau et al. 2002). In the last 50 years, many metaheuristics have been developed to adress the CVRP. For example, tabu search (TS) is the most widely used technique, which is based on a simple mechanism to prevent the process from cycling over a sequence of solutions. The TS metaheuristic can provide a good compromise between solution quality and computing time. Many researchers have proposed efficient variants of the standard TS algorithm and the best known results belong to Taillard (1993) and Rochat and Taillard (1995). The UTS algorithm of Cordeau et al. (2002) and other authors had similar results by using Tabu Search, such as (Osman 1993; Gendreau et al. 1994; Toth and Vigo 2003). Simulated annealing (SA) has also gained the similar results (Osman 1993). TS and SA are local search-based metaheuristics. At each iteration, they select the best solution in the neighborhood of the current solution as the new current solution. Their performances thus highly depends on the choice of a good initial solution. The large sized VRP can be efficiently solved by using variable neighborhood search method (Chen et al. 2010). Mester and Bräysy (2007) proposed an adaptation of the active guided evolution strategies metaheuristic that gives highly competitive results. Tarantilis (2005) presented an adaptive memory programming method for the CVRP. Marinakis (2012) proposed a modified version of greedy randomized adaptive search procedure that is rather good at computation time.

Over the past decade, a number of nature-inspired metaheuristics have been proposed to address the VRP. The most commonly used methods are genetic algorithms. Baker and Ayechew (2003) proposed a competitive GA. Berger and Barkaoui (2004) introduced a hybrid GA which combines evolutionary search and local search. Prins (2004) has developed an algorithm combining two main features of evolutionary search, namely crossovers and mutations. And then improvements are obtained by a local search procedure applied to a candidate solution. A good optimized crossover genetic algorithm was introduced recently by Nazif and Lee (2012). Particle swarm optimization (PSO) is a population-based search method that mimics the behavior of group organism as a searching method. PSO was also proposed for solving the CVRP (Chen et al. 2006; Ai and Kachitvichyanukul 2009; Marinakis et al. 2010) and many other natue-inspired algorithms (such as Yu et al. 2009; Niu et al. 2015). A more detailed descriptions of algorithms for the VRP can also be found in the surveys (Laporte et al. 2000; Cordeau et al. 2002; Eksioglu et al. 2009) and in the books (Toth and Vigo 2002; Golden et al. 2008).

3 Related work

In this section, we briefly present related work on the chemical reaction optimization framework, the unified tabu search algorithm and genetic algorithms.

3.1 Chemical reaction optimization

Chemical reaction optimization (Lam and Li 2012) is a variable population-based metaheuristic. It mimics the chemical reaction process where the interactions of molecules go toward the minimum state of free energy, similar to objective function in optimization problems. A molecule (w) has potential energy (PE), kinetic energy (KE), hit numbers and other optional characteristics to represent a solution of the considered problem. The two key properties attached to a molecule are PE and KE. The former corresponds to the fitness value of the solution while the latter is used to control the toleration to new solutions having worse fitness. The CRO method implements four types of chemical reactions, including on-wall ineffective collision, decomposition, intermolecular ineffective collision and synthesis. The on-wall ineffective collision and decomposition reactions are single molecule reactions, while the inter-molecular ineffective collision and synthesis reactions are multiple molecule reactions. Exploitation is mainly produced by on-wall ineffective

collision and inter-molecular ineffective collision, while the two remaining reactions gain exploration. After doing a number of reactions, the potential energy changes to the lowest state and the best solution is the molecule with the lowest *PE*. More details about CRO can be found in (Lam and Li 2010, 2012; Lam et al. 2013).

As suggested by experts, CRO is proposed as a general optimization framework, therefore dedicated problem-specific heuristics should be integrated into the four elementary reactions. Moreover, if decomposition and synthesis operations are not well designed, they will have low efficiency. The pseudocode of basic CRO as in (Lam and Li 2012) is presented in Algorithm 1 and it is suitable to implement CRO with an object-oriented program language.

Recently, a slightly different approach from the CRO was proposed by Astudillo et al. (2015), named Chemical reaction algorithm (CRA). This algorithm is a greedy approach for the CRO paradigm. The CRA algorithm has a simpler parameter representation and is proven to be efficient. CRA employs the broad exploration mechanisms in combination with the elitist reinsertion strategy. These characteristics can reduce the probability that the CRA algorithm stagnates in local optima.

Algorithm 1 The CRO Algorithm

Input:
Objective function f, constraints and the dimensions of the problem;
Output:
The best solution found and its objective function value;

\\ Initialization
Set *PopSize, KELossRate, MoleColl, buffer, InitialKE,* α and β;
Create *PopSize* number of molecules;
\\ Iterations
while the stopping criteria not met **do**
  Generate $b \in [0, 1]$;
  if $b >$ *MoleColl* **then**
    Randomly select one molecule M_w;
    if Decomposition criterion met **then**
      Trigger *Decomposition*;
    else
      Trigger *On-wallIneffectiveCollision*;
    end if
  else
    Randomly select two molecules M_{w1} and M_{w2};
    if Synthesis criterion met **then**
      Trigger *Synthesis*;
    else
      Trigger *Inter-molecularIneffectiveCollision*;
    end if
  end if
  Check for any new minimum solution;
end while
\\ The final stage
Output the best solution found and its objective function value;

The chemical reaction optimization was shown to outperform many existing evolutionary algorithms. It has been successfully applied to the quadratic assignment problem (Lam and Li 2010), channel assignment problem in wireless mesh network (Lam and Li 2012), traveling salesman problem (Sun et al. 2011), knapsack problem (Truong et al. 2013, 2015), heterogeneous computing environments (Li et al. 2012), scheduling scheme on heterogeneous computing systems (Xu et al. 2013, 2015), network coding optimization (Pan et al. 2011), modular neural networks applied in emotion classification (Sánchez et al. 2014), fuzzy controller design for mobile robots (Astudillo et al. 2013; Melin et al. 2013; de la Castillo et al. 2015), producing a hybrid method for optimization (Nguyen et al. 2014), and many other problems.

3.2 Unified tabu search algorithm

The unified tabu search algorithm (Cordeau et al. 2001) is a local search metaheuristic, which was initially designed for the periodic VRP and the multi-depot VRP. The UTS algorithm was then later modified to solve the CVRP (Cordeau et al. 2002) and other VRP with time windows variants. UTS has the ability to explore infeasible solutions during its search. Let S denote the set of solutions, a solution $s \in S$ may violate the constraints of the VRP. Therefore, the authors use self-adjusting positive coefficients α, β and γ for violation of vehicle capacity, route duration and customer service time windows constraints, respectively. The UTS starts from a given solution s and chooses, at each iteration, the best non-tabu solution $\bar{s}$ in the neighborhood $N(s)$. If $\bar{s}$ does not violate capacity constraint, the value of α is divided by a factor $1 + \delta$; else it is multiplied by that factor. The same rule applies to β and γ. The best value reported by the authors is $\delta = 0.5$. The diversification is implemented as follows: any solution $\bar{s} \in N(s)$ that has $\text{fit}(\bar{s}) \geq \text{fit}(s)$ is penalized by a factor $p(\bar{s})$. Finally the best solution s^* found by the search is post-optimized. The pseudocode of UTS algorithm, as in (Cordeau et al. 2002), is presented in Algorithm 2.

To our knowledge, the UTS algorithm has been successfully applied to many variations of the VRP. It is relatively simple and flexible. The UTS ranks high on accuracy and quite well on speed using just a simple mechanism. It may be the simplest of all tabu search implementations for the VRP. However, the UTS algorithm is not the best available for the VRPs and it does not always yield a solution that minimize the number of vehicles used.

3.3 Genetic algorithm

Genetic algorithms were proposed by Holland (1975) and the principles of GAs are well known. GAs are inspired by the natural selection process where a population consisting of solutions encoded as bitstrings or chromosomes will gradually improve through a *selection*, *crossover* and *mutation* process. The chromosomes with higher fitness val-

Algorithm 2 The UTS Algorithm

Input:
Solution s;
Output:
The best solution found s^*;
1: Set $\alpha = 1$, $\beta = 1$ and $\gamma = 1$;
2: **if** s is feasible **then**
3: $s^* = s$;
4: $c(s^*) = c(s)$;
5: **else**
6: $c(s^*) = \infty$;
7: **end if**
8: **for** $k = 1, \ldots, \eta$ **do**
9: Choose a solution $\bar{s} \in N(s)$ that minimizes $f(\bar{s}) + p(\bar{s})$ and is not tabu or satisfies the aspiration criteria;
10: **if** solution $\bar{s}$ is feasible and $c(\bar{s}) < c(s^*)$ **then**
11: $s^* = \bar{s}$;
12: $c(s^*) = c(\bar{s})$;
13: **end if**
14: Compute $q(\bar{s})$, $d(\bar{s})$ and $w(\bar{s})$ and update α, β and γ accordingly;
15: $s = \bar{s}$;
16: **end for**
17: Apply a post-optimization heuristic to each route of s^*;
18: **return** s^*;

ues are more likely to survive while the ones with lower fitness values are more likely to disappear. GAs rely on three basic operators: selection, crossover and mutation. The selection operator simulates the natural selection to select chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce. The crossover operator employs the selection procedure to take two parents from the population and then exchanges some of their parts to reproduce the offspring solutions. For example, in the case of a bitstring encoding, the classical one-point crossover selects a cut point on the two parent strings and then exchanges their end parts. The offspring may have higher quality. To prevent the optimization process from getting trapped into local optima, GAs use the mutation operator on offspring, then replace the worst citizen of the population by the resulting offspring. Mutation operator is considered as a secondary operator that processes each offspring position by position and flips the bit value at each position with a small probability. At the end of each iteration, the offspring together with the solutions from the previous generation form a new generation, after undergoing a selection process to keep a constant population size. Starting from a randomly or heuristically generated initial population, this cycle is repeated for a number of generations, and the best solution found is returned at the end.

cThe basic GA is very generic, and there are many aspects that can be implemented differently according to the specific problem such as the representation of solutions or chromosomes, the selection strategy, and the type of crossover and mutation operators. When applied to the vehicle routing problems, the basic GA scheme is often modified. In particular, the encoding of solutions into chromosomes is either ignored by applying the various operators directly on the solutions or designed in a very particular way to get advantages of specialized crossover and mutation operators (Baker and Ayechew 2003; Nazif and Lee 2012). More specifically, the solutions are encoded using integer strings of fixed length with or without route delimiters, while bitstrings are used in classical GAs. In this case, an integer stands for a vertex and the sequence of integers in the solution string is the order that customers are serviced. With this representation, a straightforward application of a classical GA does not work. For example, with the classical one-point crossover, the offspring may have some vertices are duplicated while others may be missed. The offspring may also violate some of the problem requirements and a repair phase is required. Therefore, numerous studies have proposed specialized operators which allow valid offspring routes to be generated. Moreover, to gain better performance, GA has been combined with local search heuristics (Berger and Barkaoui 2004; Prins 2004). However, GAs are still slower than many tabu search algorithms, it is only faster than some of them. In general, GAs are adaptable, robust and have become one of the most popular approach for designing evolutionary algorithms. GAs have been successfully applied to a huge number of applications in many fields of optimization.

4 Design of the CROUTS

The CRO framework has three stages: initialization, iteration and the final stage. The initial population, initial values of parameters *KElossRate*, *InitialKE*, *PopSize*, *MoleColl* and *buffer* are generated in the first stage. The second stage simulates the process of reacting. In this stage, according to the condition, there are four types of elementary reactions occurring. They are *on-wall ineffective collision*, *decomposition*, *inter-molecular ineffective collision* and *synthesis*. The two first operators are uni-molecular collisions, the rest are inter-molecular collisions. Decomposition and synthesis act as global search operators while on-wall ineffective collision and inter-molecular ineffective collision are local search operators. This section presents the infrastructure and rationale of the proposed CROUTS algorithm.

4.1 Solution representation

A molecule corresponds to one solution of the CVRP, and has the following characteristics: *PE*, *KE*, *InitialKE*, *minHits*, *numHits*, *MoleColl*. *PE* is positive and is the total route length of this solution. To represent the routes and the sequence of customers serviced in each route, we adapt the permutation

of n integer string which contains both customers and route splitters as shown in the following solution example for a 7-customer VRP with three routes.

Route No. 1: $0 \rightarrow 3 \rightarrow 7 \rightarrow 2 \rightarrow 0$.
Route No. 2: $0 \rightarrow 1 \rightarrow 4 \rightarrow 0$.
Route No. 3: $0 \rightarrow 5 \rightarrow 6 \rightarrow 0$.

The coded string is

$$-3 \rightarrow 7 \rightarrow 2 \rightarrow -1 \rightarrow 4 \rightarrow -5 \rightarrow 6.$$

In the coded string of this example, the numbers are the customer indexes. The $(-)$ sign standing in front of a customer index indicates that the customer is the first one of the route. In this example, route 1 begins at the depot, visits customer 3, 7 and 2 in that order then returns to the depot, and so on. This representation is unique and one string can only be decoded to one solution. The $(-)$ sign plays the role of route delimiter.

4.2 Unified tabu search adaptation

In order to yield better search ability, we allow the infeasible during the search by relaxing the constraints on the maximum vehicle load and route duration. Given a solution s, the objective value will be calculated as $f(s) = c(s) + \alpha_1 q(s) + \alpha_2 d(s)$ where $c(s)$ is total travel cost of its routes, $q(s)$ and $d(s)$ is respectively the total violation of vehicle capacity and duration, α_1 and α_2 are penalty parameters which are adjusted dynamically during the search. The authors Cordeau et al. (2001) update these penalty parameters based only on the current solution. We instead employ information of the complete population to update them. Let $\bar{q}$ and $\bar{d}$ be the average violation of vehicle capacity and route duration all over the current population. Let

$$h = \begin{cases} c(s_{\text{worst}}) & \text{if there is no} \\ & \text{feasible solution in the population;} \\ c(s_{\text{best feasible}}) & \text{otherwise.} \end{cases}$$

Then the two parameters are computed by the following equations:

$$\alpha_1 = h\frac{\bar{q}}{\bar{q}^2 + \bar{d}^2}; \quad \alpha_2 = h\frac{\bar{d}}{\bar{q}^2 + \bar{d}^2} \tag{1}$$

Whenever the algorithm finds a new best feasible solution, h is recalculated then all fitness values will be recomputed using the updated penalty parameters. Finally, the best solution found will be post-optimized by 2-opt (Lin 1965) and CROSS-exchange (Taillard et al. 1997) operators.

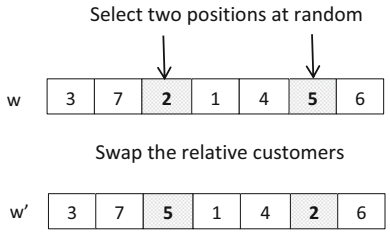

Fig. 1 Single swap mutation operator

4.3 Elementary chemical reaction operators

This subsection is dedicated to present four elementary chemical reaction operations designed for the proposed algorithm. They are on-wall ineffective collision, decomposition, synthesis and inter-molecular ineffective collision.

4.3.1 On-wall ineffective collision operator

An on-wall ineffective collision occurs when a molecule hits a wall of the container, and then bounces back. Suppose that the source molecular structure is w. Then, after the reaction we obtain a new molecule w' from the neighborhood of w. We adopt the single swap mutation operator of GAs here. An illustration of this mutation operator is presented in Fig. 1. Two customers in w will be chosen randomly, and are then exchanged. After that, the resulting solution may be feasible or infeasible. However, it will be educated by the adapted UTS to improve its quality, in all cases. The pseudocode of the on-wall ineffective collision operator is described in Algorithm 3.

Algorithm 3 Onwall(w)

Input:
Molecule w;
Output:
New molecule w';
1: Duplicate w to generate w';
2: Generate i, j randomly in $[1, n]$;
3: Swap $w'(i)$ and $w'(j)$;
4: UTS(w');
5: return w';

4.3.2 Decomposition operator

When a molecule hits a wall of the container too vigorously, it will be decomposed into two pieces and the resultant molecules should be very different from the original one. We adopt the order crossover (OX) operator of GAs to implement decomposition because OX is better suited for cyclic permutations and it is fast. The first parent is the original molecule w, and the second one is generated randomly. The

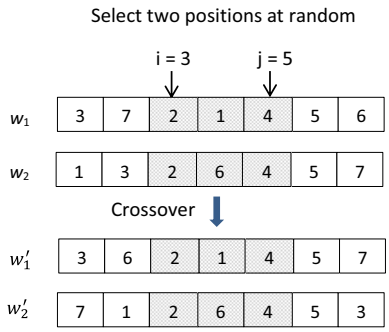

Fig. 2 Example of OX crossover

example in Fig. 2 shows how OX constructs the offspring. Firstly, two cutting points i and j are randomly selected. In this example, we have $i = 3$ and $j = 5$. Then, the substring between i and j of the first parent w_1 is copied into offspring w'_1. Finally, the second parent w_2 is swept circularly from $j+1$ onward to complete the first offspring w'_1 with the missing customers. The first offspring w'_1 is filled circularly from $j + 1$, too. The roles of parents are then exchanged to obtain the second offspring. The adapted UTS is also used to ensure that the constraints are met before returning results. The pseudocode of the decomposition operator is presented in Algorithm 4.

Algorithm 4 Decomposition(w)

Input:
Molecule w;
Output:
Two new molecules w'_1 and w'_2;
1: Duplicate w to generate w_1;
2: Generate w_2 randomly;
3: Generate i, j randomly in $[1, n]$;
4: Copy part $w_1[i, \ldots, j]$ to $w'_1[i, \ldots, j]$;
5: Copy part $w_2[i, \ldots, j]$ to $w'_2[i, \ldots, j]$;
6: From j+1, swept w_2 onward to copy the missing customers to w'_1;
7: From j+1, swept w_1 onward to copy the missing customers to w'_2;
8: UTS(w'_1);
9: UTS(w'_2);
10: return w'_1 and w'_2;

4.3.3 Synthesis operator

The synthesis reaction is opposite to decomposition reaction, two molecules collide and then combine into one molecule. In this operator, the partially mapped crossover (PMX) of GAs is employed. Suppose that we attempt to synthesize two molecules w_1 and w_2 into a new molcule w'. The PMX operator works as follows. Firstly, the two cutting points are chosen randomly. Then, the part between these two cutting points of w_1 is copied to w', and the other remaining positions of w' are filled with the remaining customers so that their absolute positions are inherited as much as possible from w_2. After that, the resulting molecule is sent to the adapted UTS procedure to ensure that the constraints are met and to improve the solution quality. The detailed pseudocode of the synthesis operator is described in Algorithm 5.

Algorithm 5 Synthesis(w_1,w_2)

Input:
Two molecules w_1 and w_2;
Output:
The new molecule w';
1: Generate i, j randomly in $[1, n]$;
2: Copy part $w_1[i, \ldots, j]$ to $w'[i, \ldots, j]$;
3: **for** $k = i$ to j **do**
4: **if** ($geneP2 = w_2[k]$ have not been copied) **then**
5: $geneP1 = w_1[k]$;
6: $indexP2j$ = index of $geneP1$ in w_2;
7: $geneP2j = w_1[indexP2j]$;
8: **if** ($geneP2j = null$) **then**
9: $w'[indexP2j] = geneP2$;
10: **else**
11: $indexP2k = w_2[geneP2j]$;
12: $w'[indexP2k] = geneP2$;
13: **end if**
14: **end if**
15: **end for**
16: Copy the remaining non-copied customers in w_2 to w';
17: UTS(w');
18: return w';

4.3.4 Inter-molecular ineffective collision operator

Inter-molecular ineffective collision represents the case where two molecules collide, and then bounce away. Suppose that two new molecules w'_1 and w'_2 are produced from two input molecules w_1 and w_2. This operator aims at minimizing the number of vehicles used and cost concurrently while checking for feasibility.

As illustrated in Fig. 3, the new solutions are generated as follows: from each parent, one route is picked randomly, then we remove the customers of picked route from the other parent. For example, the first route that contains customer 1 and customer 2 is chosen from parent w_1, these two customers are removed from parent w_2 and then they are reinserted sequentially into the best possible positions to produce child w'_2. We exchange the roles of parents to produce child w'_1. Finally, the two children are improved by the adapted UTS algorithm before returning. The pseudocode of inter-molecular ineffective collision operator is presented in Algorithm 6.

5 Experimental results

In this section, we present simulation results of the proposed algorithm. The objective of this section is twofold. The first is

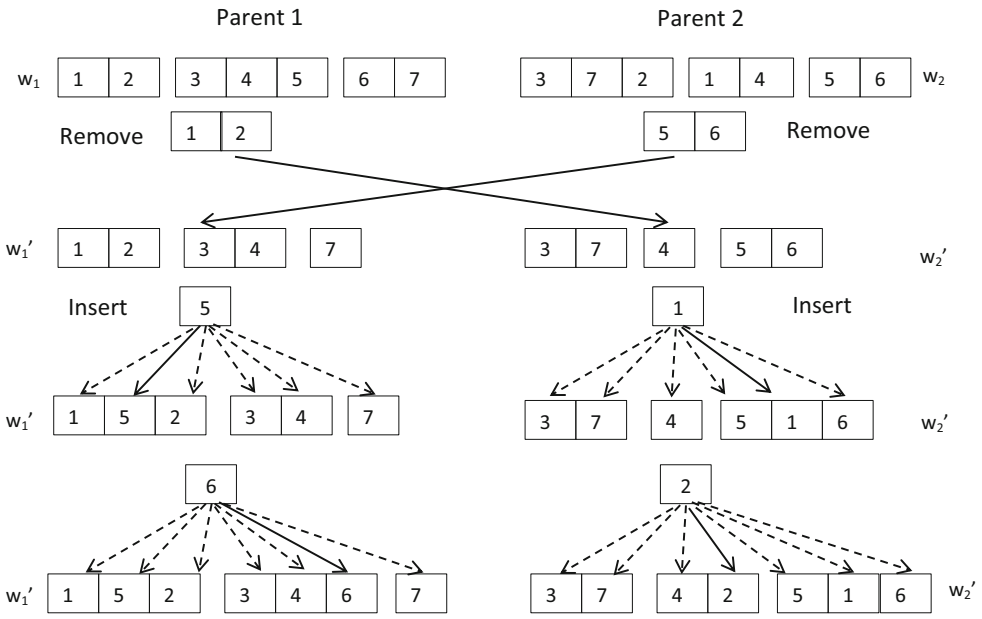

Fig. 3 Illustration of inter-molecular ineffective collision operator

Algorithm 6 Inter-moles-collide(w_1,w_2)

Input:
Two molecules w_1 and w_2;
Output:
Two new molecules w_1' and w_2';
1: Duplicate w_1 to generate w_1';
2: Duplicate w_2 to generate w_2';
3: Select route $r1$ of w_1' randomly;
4: Select route $r2$ of w_2' randomly;
5: Delete the customers of route $r1$ from w_2';
6: Delete the customers of route $r2$ from w_1';
7: Reinsert the customers of route $r1$ into w_2' in best positions;
8: Reinsert the customers of route $r2$ into w_1' in best positions;
9: UTS(w_1');
10: UTS(w_2');
11: return w_1' and w_2';

analyzing the parameter settings and the second is evaluating performance of the proposed CROUTS through a comparison against the known best-so-far results reported in the literature. The whole CROUTS algorithm was implemented in C++. All simulations were run on a personal computer equipped with an Intel Core2 Duo 2.40 GHz CPU, and 4 GB of RAM, running the Microsoft Windows 7 64 bit operating system. The CROUTS algorithm was tested with various famous CVRP benchmarks.

5.1 Parameter analysis

The parameter setting plays an important role to the efficiency of an algorithm. The CROUTS has two groups of parameters, one for the CRO and one for the UTS. With CRO, there are seven parameters (i.e. *InitialKE*, *PopSize*, *KElossRate*, *MoleColl*, *buffer*, α, β). Therefore, it is impractical to perform a complete evaluation of all combinations of these parameters. We only tackle the question how to assign parameter values that gives relatively good performance concerning both solution quality and computation time. After thorough testing and deducing from the literature (Lam and Li 2010), the selected parameters are summarized in Table 1. Note that the function evaluation limit is set to 1,000, which is less than the 100,000 value used in Lam and Li (2010). Two threshold parameters α and β are used to denote the decomposition and synthesis criteria, respectively. For uni-molecular collisions, decomposition will take place if a molecule cannot find a better solution with the number of hits larger than α, otherwise the on-wall ineffective collision happens. For inter-molecular collisions, if both molecules have kinetic energy less than β then the synthesis will be performed. Otherwise the inter-molecular ineffective collision occurs. These two parameters are tuned. We set $\alpha \in [10, 100]$, $\beta \in [10, \frac{\text{Total PE of Initial Pop}}{20}]$. Then, a number of alternative values were tested, we made 50 independent runs for each chosen pair, the ones that gave the best results were selected.

The number of iterations of the UTS algorithm was chosen in the [50, 100] range. Note that if a larger number of iterations is used, much more time could be spent to improve solutions. Experiments have also showed that appropriate parameter values change with the size of instances. Besides, the UTS procedure used a tabu list length of $\theta = 5\log_{10}(n)$, this value is smaller than the $\theta = 7.5\log_{10}(n)$ that the authors Cordeau et al. (2002) had used.

Table 1 Parameter values

Parameter	Value
KElossRate	0.8
PopSize	20
MoleColl	0.2
buffer	0
InitialKE	1000
Function evaluation limit	1000
α	[10, 100]
β	$[10, \frac{\text{Total PE of Initial Pop}}{20}]$
Number of UTS iterations	[50, 100]
Tabu list length θ	$5\log_{10}(n)$

In order to obtain a good initial population including 20 individuals, the first 10 solutions were generated by the modified Clarke-and-Wright algorithm using parameter λ (Yellow 1970), with $\lambda \in [0.5, 2]$ generated randomly for each solution, the remainders were created randomly.

5.2 Numerical results

The proposed algorithm was evaluated on two sets of the CVRP benchmarks. The first set was proposed by Christofides et al. (1979), and the second set was the same sixteen benchmark instances that had been used by Chen et al. (2006) and Ai and Kachitvichyanukul (2009). The first set consists of fourteen benchmark instances and the number of nodes varies from 51 to 200 nodes including the depot. Each instance has vehicle capacity constraints while the benchmark instances 6–10, 13 and 14 plus the restriction of maximum route lengths and the existence of nonzero service times. In the second set, the total number of nodes varies from 33 to 135 nodes including the depot, and there exist vehicle capacity constraints and a predefined minimal number of vehicles. The efficiency of algorithms for the VRP is measured by the quality of their produced solutions. This quality is expressed in terms of the relative deviation from the best-so-far (BSF) (or the best known) solution to date reported in the literature, that is:

$$dev = \frac{c_{\text{CROUTS}} - c_{\text{BSF}}}{c_{\text{BSF}}} \times 100\,(\%) \tag{2}$$

where c_{CROUTS} denotes the cost of the solution found by the proposed algorithm and c_{BSF} is the cost of the best-so-far solution.

Concerning the first set, some of the best techniques have been selected, namely the unified tabu search (UTS) algorithm of Cordeau et al. (2002), the particle swarm optimization (PSO) algorithm of Ai and Kachitvichyanukul (2009), the hybrid genetic particle swarm optimization (HybPSO) algorithm of Marinakis et al. (2010) and two GA methods proposed by Berger and Barkaoui (2004) (denoted as GAB) and by Nazif and Lee (2012) (the optimized crossover genetic algorithm, denoted as OCGA). Results related to the best objective functions found by these algorithms on Christofides et al. benchmarks are reported in Table 2. In the table, the first column describes the instance number. The next five columns provide the best results presented by the compared algorithms. The seventh, eighth and ninth columns give the best result, relative deviation and standard deviation of the proposed algorithm. The last column lists the best-so-far solutions previously reported in the literature. Row *Number* presents the number of best known results that has been found by the corresponding method; row *AvgDev* gives the average deviation from the best known solution values over all 14 instances. The bold values on the tables mean that the best known solution has been found. For this classical set, the best result belongs to Rochat and Taillard (1995) with the average deviation is 0. It can be seen from Table 2 that our proposed algorithm has reached the best known solution for ten out of the fourteen benchmark instances. For the rest of the instances, the solution quality is between 0.5266 and 0.7164 %. A close observation of Table 2 shows that CROUTS slightly outperforms OCGA and HybPSO in some instances.

Computation time is as important as solution quality in solving optimization problems. In fact, we often aim at establishing a balanced tradeoff between solution quality and computation speed. Average computation times (in minutes) as well as used computers for the first set are reported in Table 3 where the first column also describes the instance number, the second column shows the number of nodes and the remaining columns provide average wallclock times of the algorithms. The computation times for instances C4, C5, C9, C10, C11 and C13 of the OCGA (Nazif and Lee 2012) are not reported. Our average accumulated computing time per instance is 1.99 min. This number indicates that the proposed algorithm is quite fast even the computer differences are considered. Regarding the computational efficiency, for the first benchmark set, the CROUTS produces high-quality solutions within an acceptable time.

Concerning the second set, the hybrid Discrete Particle Swarm Optimization algorithm (DPSO) of Chen et al. (2006) and the PSO of Ai and Kachitvichyanukul (2009) have been selected for comparison. This benchmark set includes 16 instances, with the number of nodes varies from 33 to 135 nodes including the depot, vehicles only have capacity constraints and the total number of vehicles varies from 3 to 10 vehicles. More specifically, this set is comprised of 9 instances of Augerat et al. (3 instances begin with A, 4 instances begin with B and 2 instances begin with P), 2 instances of Christofides and Eilon beginning with E, 3

Table 2 Comparison of results on benchmarks of Christofides et al.

Instance	UTS	GAB	OCGA	PSO	HybPSO	CROUTS			BSF
						Best	Dev	SD	
C1	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	0.0000	0	524.61[a]
C2	835.45	**835.26**	**835.26**	844.42	**835.26**	**835.26**	0.0000	0	835.26[a]
C3	829.44	827.39	**826.14**	829.40	**826.14**	**826.14**	0.0000	1.34	826.14[a]
C4	1038.44	1036.16	**1028.42**	1048.89	1028.42	1034.03	0.5455	2.75	1028.42[a]
C5	1305.87	1324.06	1299.64	1323.89	1291.45	1298.09	0.5266	4.64	1291.29[b]
C6	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	**555.43**	0.0000	0	555.43[a]
C7	**909.68**	**909.68**	**909.68**	917.68	**909.68**	**909.68**	0.0000	0	909.68[a]
C8	866.38	868.32	**865.94**	867.01	868.45	**865.94**	0.0000	2.15	865.94[a]
C9	1171.81	1169.15	1163.38	1181.14	1164.35	1169.38	0.5875	3.79	1162.55[a]
C10	1415.40	1418.79	1406.23	1428.46	1396.18	1405.85	0.7164	2.58	1395.85[b]
C11	1074.13	1043.11	**1042.11**	1052.34	1044.03	**1042.11**	0.0000	1.56	1042.11[a]
C12	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	0.0000	0	819.56[a]
C13	1568.91	1553.12	1542.25	1546.20	1544.18	**1541.14**	0.0000	2.27	1541.14[a]
C14	866.53	**866.37**	**866.37**	**866.37**	**866.37**	**866.37**	0.0000	0	866.37[a]
Number	4	6	10	4	7	10			
AvgDev	0.69 %	0.48 %	0.11 %	0.88 %	0.084 %		0.17 %		

[a] Taillard (1993)
[b] Rochat and Taillard (1995)

Table 3 Comparison of computation times on benchmarks of Christofides et al.

Instance	Nodes	UTS[a]	GAB[b]	OCGA[c]	PSO[d]	HybPSO[e]	CROUTS[f]
C1	51	4.57	2	0.81	0.40	0.05	0.40
C2	76	7.27	14.33	3.92	0.95	0.21	0.68
C3	101	11.23	27.90	8.01	1.68	0.32	1.51
C4	151	18.72	48.98	–	3.72	1.01	0.03
C5	200	28.10	55.41	–	6.88	2.15	4.63
C6	51	4.61	2.33	1.25	0.50	0.05	0.71
C7	76	7.55	10.50	4.29	1.15	0.28	0.99
C8	101	11.17	5.05	7.35	1.92	0.89	1.48
C9	151	19.17	17.88	–	4.92	1.57	2.47
C10	200	29.74	43.86	–	8.62	3.01	6.10
C11	121	14.15	22.43	–	1.55	0.53	3.62
C12	101	10.99	7.21	4.45	1.47	0.38	3.06
C13	121	14.53	34.91	–	2.67	0.41	1.27
C14	101	10.65	4.73	6.57	1.65	0.37	0.96
Average time		13.75	21.25	–	2.72	0.80	1.99

[a] Sun Ultrasparc 10 440 MHz
[b] Pentium II 400 MHz
[c] Pentium IV 2 GHz
[d] Pentium IV 3.4 GHz
[e] Centrino Mobile Intel Pentium M750 1.86 GHz
[f] Intel Core2 Duo 2.40 GHz

instances of Fisher beginning with F and 2 instances of Christofides et al. beginning with M. The computational results in terms of the best objective function found and the average computation time of the algorithms are shown in Table 4. The first three columns indicate the instance name, the number of customers and the number of vehicles, respectively. The fourth column provides the best-so-far results reported in the literature. The next three columns present the best objective function found by the three algorithms. Columns *dev* and *stddev* respectively present the relative deviation and the standard deviation of the proposed algorithm. And the last three columns report computation times

Table 4 Comparison of computational results on benchmarks of Chen et al. and Ai and Kachitvichyanukul

Instance	Nodes	No. veh.	BSF	Cost					Computation time (s)		
				DPSO	PSO	CROUTS	Dev	SD	DPSO[a]	PSO[b]	CROUTS[c]
An33k5	33	5	661	**661**	**661**	**661**	0.0000	0	32	13	19
An46k7	46	7	914	**914**	**914**	**914**	0.0000	0	129	23	20
An60k9	60	9	1354	**1354**	1355	**1354**	0.0000	1.74	309	40	28
Bn35k5	35	5	955	**955**	**955**	**955**	0.0000	0	38	14	13
Bn45k5	4	5	751	**751**	**751**	**751**	0.0000	0	134	20	20
Bn68k9	68	9	1272	**1272**	1274	1274	0.1572	1.54	344	50	35
Bn78k10	78	10	1221	1239	1223	1223	0.1683	3.84	429	64	60
En30k3	30	3	534	**534**	**534**	**534**	0.0000	0	28	16	10
En51k5	51	5	521	528	**521**	**521**	0.0000	0	301	22	20
En76k7	76	7	682	688	**682**	**682**	0.0000	0.68	527	60	37
Fn72k4	72	4	237	244	**237**	**237**	0.0000	0	398	53	42
Fn135k7	135	7	1162	1215	**1162**	**1162**	0.0000	1.16	1526	258	207
Mn101k10	101	10	820	824	**820**	**820**	0.0000	0	874	114	93
Mn121k7	121	7	1034	1038	1036	**1034**	0.0000	1.29	1734	89	79
Pn76k4	76	4	593	602	594	**593**	0.0000	0.96	496	48	41
Pn101k4	101	4	681	694	683	**681**	0.0000	0	978	86	80
Number				7	10	14					
Average				0.97 %	0.07 %		0.02 %		517.31	60.63	47.75

[a] Mobile Intel Pentium IV CPU 1.80 GHz
[b] Pentium IV 3.4 GHz
[c] Intel Core2 Duo 2.40 GHz PC

Table 5 Results of the Friedman test ($\alpha = 0.05$)

Benchmark set	Friedman value	Value of χ^2	p value
Benchmarks of Christofides et al.	18.6429	9.2959	<0.00001
Benchmarks of Chen et al.	7.3256	5.9063	0.00257

in second of the algorithms. It can be easily seen in this table that the proposed algorithm is highly efficient on this set as it reaches the best-so-far solution for fourteen of the sixteen instances, and for the remainders the deviation is very small (0.1572 % for instance Bn68k9 and 0.1638 % for instance Bn78k10). The average accumulated computing time per problem is 47.75 s. This number suggests that the proposed method is fast, at least not inferior to other heuristics even when the computer differences are considered.

The first benchmark set contains larger problems than the second set, and it should be noted that research on addressing the problem of larger size can be a challenging task. Nevertheless, computational results suggest that the proposed algorithm is able to find high-quality solutions for benchmark instances of this set within a reasonable time. The proposed algorithm produces better results than UTS and GA on large instances C5 and C10 of the first benchmark set. The average deviations from the best known solutions are 0.17 and 0.02 % for the first and the second set of benchmarks, respectively.

Furthermore, to perform a multiple comparison, it is necessary to check whether all the results obtained by the algorithms present any inequality. Therefore, statistical tests were also performed. The chosen method is the Friedman test (Sheskin 2007), which is a nonparametric analogue of the parametric two-way analysis of variance by ranks. Table 5 shows the result obtained by applying the Friedman test on the results of the two benchmark sets, to see whether there are global differences in the results. In this table, the p values of the Friedman test are lower than the level of significance considered $\alpha = 0.05$, indicating that there are significant differences among the observed results produced by the algorithms. Tables 6 and 7 summarize the ranking obtained by the Friedman test on the results of the two benchmark sets, respectively. The best performing algorithm is the algorithm that obtains the lowest rank as computed by the Friedman test. According to the results obtained, the proposed CROUTS algorithm is ranked second on the benchmarks of Christofides et al. and is ranked first on the benchmarks of Chen et al.

Table 6 Rankings obtained through Friedman test on the benchmarks of Christofides et al.

Algorithm	Mean rank
UTS	4.5714
GAB	3.8929
OCGA	2.5
PSO	4.5714
HybPSO	2.8571
CROUTS	2.6071

Table 7 Rankings obtained through Friedman test on the benchmarks of Chen et al.

Algorithm	Mean rank
DPSO	2.4678
PSO	1.9063
CROUTS	1.625

The high-quality results yielded by the proposed algorithm may be explained as follows. By analyzing the chemical reaction operations designed in CROUTS and the operational environment of CROUTS, it is clear that CROUTS absorbs the advantages of both GAs and UTS. For example, the intermolecular ineffective collision of CROUTS exchanges the partial structure of two different molecules, and thus acts like the crossover operation used in GAs. The design of this operator also plays an important role to the effective of the proposed method on the second benchmark, where the number of used vehicles must be minimized. The designed on-wall collision operator of CROUTS randomly changes the molecular structures, which has an effect similar to the mutation operation of GAs. Besides, the diversification of the proposed method is enriched by using the adapted UTS to repair and improve the quality of the result molecules. In addition to the similarity to GAs and UTS, CROUTS has two additional operations: decomposition and synthesis. These two operations may change the number of individuals in a population, which will bring more opportunities to CROUTS for jumping out of local optima as well as exploring wider areas in the solution space. This characteristic also enables CROUTS to find good solutions in less time than GAs do.

6 Conclusion

In this study, we proposed a nature-inspired algorithm named CROUTS based on chemical reaction optimization and unified tabu search, to effectively solve the vehicle routing problem. The VRP is an important problem in the field of distribution and logistics where the research and applications related to the application of artificial intelligence are now emerging. One of the main contributions of this paper is to indicate that chemical reaction optimization can be used in combination with other metaheuristics to tack the VRP with remarkable results in terms of solution quality and computational efficiency. The second contribution is the utilization of crossover and mutation operators for the design of CRO operators to implement the local search and global search. Four problem-specific elementary reactions need to be carefully designed to achieve efficiency. The adaptive penalty schema is utilized for the unified tabu search algorithm to repair and improve the result solutions within each CRO operator. Finally, the proposed algorithm is flexible and could be extended to solve other VRP variants with more constraints. The algorithm was applied on two set of benchmark instances and gave competitive results. More specifically, on the classic benchmark instances of Christofides, the average quality is 0.17 % and it is 0.02 % on the second set of benchmark instances. However, the proposed algorithm has not yet overcome a drawback of the CRO method of having slightly much parameters. In the future, we plan to intensively study the influence of parameter values to enhance the performance of the proposed approach, and possibly reduce the number of parameters. Furthermore, in practice, there exists very large scale VRPs, having much more constraints. In the literature, variable neighborhood search and parallelization have shown to be efficient ways of handling large scale VRPs. Therefore, extension of the proposed approach in combination with these strategies for addressing realistic large scale VRPs is also an interesting topic for future work.

Acknowledgements The authors are grateful to the anonymous reviewers and the editors for their comments which have helped to improve the manuscript. This study was funded by the National Natural Science Foundation of China (Grant Nos. 61133005, 61432005, 61370095, 61472124, 61202109, and 61472126) and the International Science & Technology Cooperation Program of China (Grant Nos. 2015DFA11240, 2014DFBS0010). T-L. Dam was also partially supported by science research fund of Hanoi University of Industry, Hanoi, Vietnam.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Ai TJ, Kachitvichyanukul V (2009) Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. Comput Ind Eng 56(1):380–387

Astudillo L, Melin P, Castillo O (2013) Nature inspired chemical optimization to design a type-2 fuzzy controller for a mobile robot. In: IFSA world congress and NAFIPS annual meeting (IFSA/NAFIPS), 2013 Joint, pp 1423–1428

Astudillo L, Melin P, Castillo O (2015) Introduction to an optimization algorithm based on the chemical reactions. Inf Sci 291:85–95

Baker BM, Ayechew MA (2003) A genetic algorithm for the vehicle routing problem. Comput Oper Res 30(5):787–800

Berger J, Barkaoui M (2004) A new hybrid genetic algorithm for the capacitated vehicle routing problem. J Oper Res Soc 54(12):1254–1262

Chen AL, Yang GK, Wu ZM (2006) Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J Zhejiang Univ Sci A 7(4):607–614

Chen P, Huang HK, Dong XY (2010) Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. Exp Syst Appl 37(2):1620–1627

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (eds) Combinatorial optimization, vol 1. Wiley, Chichester, pp 315–338

Cordeau J, Laporte G, Hautes E, Commerciales E, Gerad L (2001) A unified tabu search heuristic for vehicle routing problems with time windows. J Oper Res Soc 52:928–936

Cordeau J, Gendreau M, Laporte G, Potvin J, Semet F (2002) A guide to vehicle routing heuristics. J Oper Res Soc 53(5):512–522

Dantzig G, Ramser J (1959) The truck dispatching problem. Manag Sci 6(1):80–91

de la O D, Castillo O, Meléndez A, Melin P, Astudillo L, Sánchez C, (2015) Optimization of reactive fuzzy controllers for mobile robots based on the chemical reactions algorithm. In: Melin P, Castillo O, Kacprzyk J (eds) Design of intelligent systems based on fuzzy logic, neural networks and nature-inspired optimization, studies in computational intelligence, vol 601. Springer, New York, pp 253–266

Eksioglu B, Vural AV, Reisman A (2009) The vehicle routing problem: a taxonomic review. Comput Ind Eng 57(4):1472–1483

Gendreau M, Hertz A, Laporte G (1994) A tabu search heuristic for the vehicle routing problem. Manag Sci 40(10):1276–1290

Golden BL, Raghavan S, Wasil EA (eds) (2008) The vehicle routing problem: latest advances and new challenges. Springer, Berlin

Holland J (1975) Adaptations in natural and artificial systems. University of Michigan Press, ann arbor

Lam AY, Li VO (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14(3):381–399

Lam AY, Li VO (2012) Chemical reaction optimization: a tutorial. Memet Comput 4(1):3–17

Lam AY, Li VO, Xu J (2013) On the convergence of chemical reaction optimization for combinatorial optimization. IEEE Trans Evol Comput 17(5):605–620

Laporte G (2009) Fifty years of vehicle routing. Transp Sci 43(4):408–416

Laporte G, Gendreau M, Potvin J, Semet F (2000) Classical and modern heuristics for the vehicle routing problem. Int Trans Oper Res 7(4–5):285–300

Li K, Zhang Z, Xu Y, Gao B, He L (2012) Chemical reaction optimization for heterogeneous computing environments. In: ISPA '12 proceedings of the 2012 IEEE 10th international symposium on parallel and distributed processing with applications. IEEE Computer Society, Washington, ISPA '12, pp 17–23

Lin S (1965) Computer solutions of the traveling salesman problem. Bell Syst Tech J 44(10):2245–2269

Marinakis Y (2012) Multiple phase neighborhood search-GRASP for the capacitated vehicle routing problem. Expert Syst Appl 39(8):6807–6815

Marinakis Y, Marinaki M, Dounias G (2010) A hybrid particle swarm optimization algorithm for the vehicle routing problem. Eng Appl Artif Intell 23(4):463–472

Melin P, Astudillo L, Castillo O, Valdez F, Garcia M (2013) Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm. Expert Syst Appl 40(8):3185–3195

Mester D, Bräysy O (2007) Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Comput Oper Res 34(10):2964–2975

Nazif H, Lee LS (2012) Optimised crossover genetic algorithm for capacitated vehicle routing problem. Appl Math Model 36(5):2110–2117

Nguyen TT, Li Z, Zhang S, Truong TK (2014) A hybrid algorithm based on particle swarm and chemical reaction optimization. Expert Syst Appl 41(5):2134–2143

Niu Y, Wang S, He J, Xiao J (2015) A novel membrane algorithm for capacitated vehicle routing problem. Soft Comput 19(2):471–482

Osman I (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann Oper Res 41(4):421–451

Pan B, Lam A, Li V (2011) Network coding optimization based on chemical reaction optimization. In: Global telecommunications conference (GLOBECOM 2011), 2011 IEEE, pp 1–5

Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Comput Oper Res 31(12):1985–2002

Rochat Y, Taillard E (1995) Probabilistic diversification and intensification in local search for vehicle routing. J Heuristics 1(1):147–167

Sánchez C, Melin P, Astudillo L (2014) Chemical optimization method for modular neural networks applied in emotion classification. In: Castillo O, Melin P, Pedrycz W, Kacprzyk J (eds) Recent advances on hybrid approaches for designing intelligent systems, studies in computational intelligence, vol 547. Springer, New York, pp 381–390

Sheskin DJ (2007) Handbook of parametric and nonparametric statistical procedures, 4th edn. Chapman & Hall/CRC Press, London/Boca Raton

Sun J, Wang Y, Li J, Gao K (2011) Hybrid algorithm based on chemical reaction optimization and lin-kernighan local search for the traveling salesman problem. In: 2011 Seventh international conference on natural computation (ICNC), vol 3, pp 1518–1521

Taillard E (1993) Parallel iterative search methods for vehicle routing problems. Networks 23(8):661–673

Taillard E, Badeau P, Gendreau M, Guertin F, Potvin JY (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. Transp Sci 31(2):170–186

Tarantilis C (2005) Solving the vehicle routing problem with adaptive memory programming methodology. Comput Oper Res 32(9):2309–2327

Toth P, Vigo D (eds) (2002) The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia

Toth P, Vigo D (2003) The granular tabu search and its application to the vehicle-routing problem. INFORMS J Comput 15(4):333–346

Truong TK, Li K, Xu Y (2013) Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem. Appl Soft Comput 13(4):1774–1780

Truong TK, Li K, Xu Y, Ouyang A, Nguyen TT (2015) Solving 0–1 knapsack problem by artificial chemical reaction optimization algorithm with a greedy strategy. J Intell Fuzzy Syst 28(5):2179–2186

Xu Y, Li K, He L, Truong TK (2013) A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization. J Parallel Distrib Comput 73(9):1306–1322

Xu Y, Li K, He L, Zhang L, Li K (2015) A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems. IEEE Trans Parallel Distrib Syst 26(12):3208–3222

Yellow PC (1970) A computational modification to the savings method of vehicle scheduling. Oper Res Q (1970–1977) 21(2):281–283

Yu B, Yang ZZ, Yao B (2009) An improved ant colony optimization for vehicle routing problem. Eur J Oper Res 196(1):171–176