Research Article

A Problem-Reduction Evolutionary Algorithm for Solving the Capacitated Vehicle Routing Problem

Wanfeng Liu^{1,2} and Xia Li^{1,2}

¹ College of Information Engineering, Shenzhen University, Shenzhen 518060, China

² Shenzhen Key Lab of Communication and Information Processing, Shenzhen 518060, China

Correspondence should be addressed to Xia Li; lixia@szu.edu.cn

Received 30 April 2014; Accepted 13 October 2014

Academic Editor: Pandian Vasant

Copyright © 2015 W. Liu and X. Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Assessment of the components of a solution helps provide useful information for an optimization problem. This paper presents a new population-based problem-reduction evolutionary algorithm (PREA) based on the solution components assessment. An individual solution is regarded as being constructed by basic elements, and the concept of acceptability is introduced to evaluate them. The PREA consists of a searching phase and an evaluation phase. The acceptability of basic elements is calculated in the evaluation phase and passed to the searching phase. In the searching phase, for each individual solution, the original optimization problem is reduced to a new smaller-size problem. With the evolution of the algorithm, the number of common basic elements in the population increases until all individual solutions are exactly the same which is supposed to be the near-optimal solution of the optimization problem. The new algorithm is applied to a large variety of capacitated vehicle routing problems (CVRP) with customers up to nearly 500. Experimental results show that the proposed algorithm has the advantages of fast convergence and robustness in solution quality over the comparative algorithms.

1. Introduction

The large-scale NP-hard problems with typically exponential complexity are difficult to solve in polynomial time. In practice, evolutionary-based algorithms are proposed for searching near-optimal solutions. Many different types of evolutionary algorithms have been proposed so far, for example, the genetic algorithms (GAs) [1], particle swarm optimization (PSO) [2], shuffled frog-leaping algorithm (SFLA) [3], memetic algorithms (MAs) [4], differential evolution (DE) [5], ant-colony optimization (ACO) [6], extremal optimization (EO) [7], and so on. Among these algorithms, ACO allocates and modifies the pheromones of each edge, and the ants search the new tours under the guidance of the pheromones; in the extremal optimization, the fitness is defined on the components of the feasible solution and the undesirable components are more liable to be eliminated. Both algorithms attempt to evaluate quantitatively the components of a solution, which can be used to guide the optimization process. They have found wide applications in many fields, and the

idea of exploring the intrinsic properties of each component in the feasible solutions forms the foundation of our work.

In this paper, a novel problem-reduction evolutionary algorithm (PREA) is proposed. The feasible solution of the problem is supposed to be composed of a series of basic elements with their respective acceptability defined. The PREA consists of the searching phase and the evaluation phase. In the evaluation phase, the acceptability of basic elements is calculated and passed to the searching phase. The searching phase attempts to search for better solutions with a group of delicately designed encapsulation processes and optimizers. Similar to the idea of “if the backbones (i.e., the easy part of problem) are held constant, the optimization process is able to concentrate on other parts, which are more difficult to solve” in searching for backbones [8], for each individual solution, the basic elements with higher acceptability (or better components) are encapsulated as a whole. The new solution is representative of a reduced-size optimization problem. Specifically, encapsulation probability is introduced into the encapsulation which helps to adjust the searching

area size of the successive optimization. The acceptability for individual solutions is also defined in the optimization phase to guide the search direction, which provides a new way for an individual to learn from others. With the evolution of the algorithm, the original optimization problem is reduced to a series of smaller-size optimization problems, thus significantly improving the convergence speed.

In this paper, the problem-reduction evolutionary algorithm is applied to the capacitated vehicle routing problem (CVRP). The CVRP proposed by Dantzig and Ramser [9] is an extension of the well-known NP-hard traveling salesman problem (TSP) and has been studied extensively in the literature. It is reported that the best exact algorithm can solve instances involving approximately 100 customers [10]. Researchers mainly focus on heuristics which can find near-optimal solutions in acceptable time. The variable neighborhood method and the simple iterated local search (ILS) are commonly used for CVRP. Li et al. [11] proposed a VRTR algorithm which combines the record-to-record (RTR) principle with a variable length neighbor list. Chen et al. [12] presented an ILS algorithm together with variable neighborhood descent based on multioperator optimization. Subramanian et al. [13] designed an ILS-based heuristic incorporating a set partitioning (SP) approach. Sequences of SP models which represent routes found by a metaheuristic approach are solved by a mixed integer programming (MIP). To overcome the limitations of a single method, hybrid algorithms have been extensively proposed; for example, Luo et al. [14] presented an improved shuffled frog leaping algorithm (SFLA) combined with the power-law extremal optimization (τ -EO), Nagata and Bräysy [15] put forward a memetic algorithm (MA) with edge assembly (EAX) crossover in the local search, and an efficient modification algorithm is applied to address the constraint violation of the infeasible solutions. Some techniques have been put forward to reduce the computational complexity of an algorithm. Zachariadis and Kiranoudis [16] presented a penalized static move descriptors algorithm (PSMDA), in which a static move descriptor (SMD) data structure is constructed to reduce the computational cost for evaluating the solution neighborhoods. Liu and Li [17] provided a fast feasibility evaluation of solution neighborhoods by introducing the concepts of “preload” and “postload.” In this paper, we propose a new evolutionary algorithm in which the evolution of both individual solutions and reduced problems is considered. Experimental results show that it is computationally efficient and claim to find new best known solutions to 7 well-studied CVRP instances.

The rest of the paper is organized as follows. Section 2 briefly described the capacitated vehicle routing problems with the novel coding mechanism for the feasible solution. In Section 3, the problem-reduction evolutionary algorithm is presented in detail. Experimental results and analysis are reported in Section 4, followed by the conclusion in Section 5.

2. The Capacitated Vehicle Routing Problem

The capacitated vehicle routing problem (CVRP) aims to find minimum total cost routes for a fleet of vehicles to serve the

given customers with known locations and demands, subject to the constraint that vehicles assigned to the routes must carry no more than a fixed quantity of goods. For the capacity and distance VRP (CDVRP), the duration of each route must not exceed an upper bound D .

In this paper, the variable length codes (VLC) are used. A feasible solution for the N -customers CVRP is expressed as an integer sequence:

$$S_i = \{s_{i1}, s_{i2}, \dots, s_{iM}\}$$

$$\text{with } \begin{cases} s_{ij} = 0 & j = 1 \text{ or } M \\ s_{ij} = 0, 1, \dots, N & \text{otherwise,} \end{cases} \quad (1)$$

where M is the coding length defined later. Equation (1) implies that the first and the last elements of S_i represent the central depot (denoted by 0), and the other elements in between can be either the depot or any customer (denoted by its index $1, 2, \dots, N$) which appears only once. For example, $S_i = \{0, 1, 3, 6, 8, 0, 4, 5, 7, 2, 0\}$ is a solution of the CVRP, and there are two routes in the solution:

Route 1: 0, 1, 3, 6, 8, 0,

Route 2: 0, 4, 5, 7, 2, 0.

The coding length M is equal to $N+K+1$ with K being the number of vehicle routes. It should be noted that the number of vehicle routes is not fixed in the proposed algorithm which means that the coding length M is variable. This implies that the CVRP with the proposed coding scheme is basically a multiobjective optimization problem which needs to be solved with both the number of vehicles and the traveling costs being optimized simultaneously.

3. Problem-Reduction Evolutionary Algorithm

The PREA is a population-based evolutionary algorithm with two phases: the searching phase and the evaluation phase. During the searching phase, we start from N randomly chosen initial feasible solutions. They are fed into individual optimizers to find “good” solutions. The optimizers may be the same or not. In the evaluation phase, analysis of the components of solutions is performed based on two factors. One is whether the solution is good (with big fitness) or not; another is the frequency of the component appearing in the individual solutions. Of course, components which are found available in most of the “good” solutions should be “reliable” components. Next, the evaluation values are fed back to the searching phase to reformulate the problem through encapsulation which will be described in detail later. As a consequence, the problem is reduced to a smaller-size one and the individual solutions can be optimized by respective optimizer with the help of the attained evaluation values of components. Again, the results are sent to the evaluation phase. The two phases are carried out alternately until all individuals converge to the same solution which is supposed to be the near-optimal solution. The novel point of PREA lies in the evaluation of the components extracted from the individual solutions and the delicate design of the

FIGURE 1: The framework of PREA.

encapsulation process which reduces the optimization scale. The framework of PREA can be described in Figure 1.

3.1. Basic Element and Its Acceptability. In the evaluation phase of PREA, solutions are decomposed to smaller components. In this paper, we confine our attention to combinatorial optimization problems in which the feasible solution can be expressed as an integer sequence. To make it concrete, we have the following definitions.

Definition 1. Given an optimization problem P of size N and its set of feasible solutions $S = \{S_1, S_2, \dots\}$ with $S_i = \{s_{i1}, s_{i2}, \dots, s_{ij}, \dots, s_{iM}\}$, $S_i \in Z^M$. The collection of adjoining pair $D_i = \{(s_{ij}, s_{i(j+1)}) \mid j = 1, 2, \dots, M-1\}$ is defined as the basic element set with respect to S_i . Define $D = \cup D_i$ as the basic element set of problem P .

Definition 2. For an optimization problem P with feasible solutions $\{S_i\}_{i=1}^n$, suppose that $\{f_i\}_{i=1}^n$ denotes the respective fitness; then the normalized fitness $\tilde{f}_i$ for S_i is defined as

$$\tilde{f}_i = \frac{f_i - f_w}{\sum_{j=1}^n (f_j - f_w)}, \quad (2)$$

where f_w is the fitness of the worst solution in the population.

It can be seen from (2) that $\tilde{f}_i \in [0, 1]$ and the sum $\sum_{i=1}^n \tilde{f}_i$ is equal to 1. Now, we present the idea of the acceptability for basic elements. As the name implies, acceptability is a quantitative measure for evaluating the acceptable degree of a basic element in a group of individuals; it is defined as follows.

Definition 3. For a population-based optimization with n agents, $\{S_1, \dots, S_p, \dots, S_n\}$ is the feasible solution set with

respective basic element set $\{D_1, \dots, D_i, \dots, D_n\}$. The acceptability of a basic element $(e_1, e_2) \in D$ is defined as

$$A(e_1, e_2) = \sum_{k=1}^n \tilde{f}_k \times \delta((e_1, e_2), D_k), \quad (3)$$

$$\delta((e_1, e_2), D_k) = \begin{cases} 1 & \text{if } (e_1, e_2) \in D_k \\ 0 & \text{else.} \end{cases}$$

Obviously, $A(e_1, e_2) \in [0, 1]$. If for an arbitrary pair (e_1, e_2) , it is not the basic element for any of the solutions, then $A(e_1, e_2) = 0$; on the contrary, $A(e_1, e_2) = 1$ implies that the pair (e_1, e_2) inevitably appears to be common component for all individual solutions.

Similarly, we can define the acceptability of a feasible solution S_i by accumulating those of the individual basic elements as

$$A(S_i) = \sum_{(e_1, e_2) \in D_i} A(e_1, e_2), \quad i = 1, 2, \dots, n. \quad (4)$$

3.2. Encapsulation Process. The encapsulation process aims to reduce the size of an optimization problem with the use of the acceptability of basic elements. To be specific, for an individual solution, the basic element or a group of basic elements can be encapsulated to form a new entity, which correspondingly represents a new reduced-size problem. The critical point lies in the way to reformulate the problem. Here,

FIGURE 2: 1-1-interchange neighborhood for an encapsulation solution.

we define the encapsulation probability for the basic element (e_1, e_2) of an individual solution S_i as

$$P(e_1, e_2) = \begin{cases} \frac{(A(e_1, e_2) - A_{\text{allow}}) P_{\text{max}}}{A_{\text{max}} - A_{\text{allow}}} & \text{if } A(e_1, e_2) > A_{\text{allow}} \\ 0 & \text{else,} \end{cases} \quad (5)$$

$$A_{\text{allow}} = \alpha A_{\text{min}} + (1 - \alpha) A_{\text{max}},$$

where A_{max} and A_{min} , respectively, represent the maximum and minimum acceptability of basic elements in S_i . $P_{\text{max}} \in (0, 1]$ is a predefined encapsulation probability with the corresponding maximal acceptability A_{max} . $\alpha \in (0, 1]$ is used to adjust the acceptability threshold A_{allow} for basic elements. A large α implies that an acceptability threshold closer to A_{min} is preferred. A basic element (e_1, e_2) in S_i is encapsulated with the encapsulation probability.

It can be seen from (5) that the parameter α basically reflects the chance of basic elements to be encapsulated. With the increase of α , more basic elements will be encapsulated and vice versa. Similarly, the bigger the value of P_{max} , the more the chance for the basic elements to be encapsulated. At the beginning of the algorithm, there are fewer common basic elements among individual solutions, which implies that there is big room for improvement; increasing the values of α and P_{max} can result in a decrease in the problem scale which makes the searching process focused on more promising areas. With the improvement of individual solutions, the common basic elements in the solution population increase, and fewer parts of the individuals can be improved. On the contrary, decreasing the values of α and P_{max} can increase

the problem scale which leads to wider areas in the searching space.

In the encapsulation process, the better basic elements are encapsulated as new nodes [18] which will keep unchanged in the later optimization process. The new solution achieved by encapsulation is called the encapsulation solution. It is used as the starting point for the next-stage optimization. For example, let $S = \{0, 4, 11, 7, 6, 3, 0, 1, 9, 10, 8, 5, 2, 0\}$ be a feasible solution of the CVRP as shown in Figure 2(a). Assume that basic elements $\{1, 9\}$, $\{9, 10\}$, $\{7, 6\}$, $\{8, 5\}$, and $\{4, 11\}$ are selected to be encapsulated. Figure 2(b) illustrates the encapsulation solution with six new nodes. From Figure 2(b) we can see that each new node has a head and a tail, which is different from the original one, and the distance between two nodes also needs to be redefined. For instance, for the reduced problem shown in Figure 2(b), the distance between the new nodes ③ and ⑤ may take on a value derived from those of the original node pairs $(1, 5)$, $(1, 8)$, $(10, 5)$, and $(10, 8)$. Obviously, the reformulation of the reduced configuration is problem specific and for the vehicle routing problems, we simply enclose some parts of the route as new nodes.

By encapsulating, the original optimization problem is reduced to a smaller-size optimization problem, thus greatly speeding up the process of searching for the “better” solutions. It is worth noting that the encapsulation varies from individual to individual and also from iteration to iteration, which can be explained as exploring diverse regions at different levels under the guidance of acceptability of basic elements.

In the optimization process for the reduced problem, it is obvious that a simple neighborhood operation in the reduced problem domain usually implies a complicated local search strategy in the original problem [18]. This is illustrated in

```

function iILS(S, Sbest)
  S = LocalSearch(S), S' = S
  do
    S = Perturbation(S)
    S = LocalSearch(S)
    if (E(S, S') = 1)
      S' = S
      if (f(S) < f(Sbest))
        Sbest = S
      endif
    else
      S = S'
    endif
  while (stop criterion satisfied)
  return S, Sbest
end function

```

ALGORITHM 1: Pseudocode for the *i*ILS.

Figure 2(d), which depicts a 1-1-interchange neighborhood for an encapsulation solution. Actually, this is the 4-opt neighborhood for the original solution shown in Figure 2(c).

3.3. Problem-Reduction Evolutionary Algorithm. In the framework of PREA shown in Figure 1, there are many choices for the selection of optimizers. To make it simpler, we assume that all optimizers are of the same type and the well-known simple-structure iterated local search (ILS) [19] is used. The main idea of ILS is to start from a randomly generated solution, then to repeatedly apply local search combined with mutation to the solution, and to reserve the better ones until the stopping criterion is met. To take the advantage of the solution acceptability and facilitate the CVRP at hand, not only the better solutions but also the worse solutions with bigger acceptability are reserved during the implementation of the ILS, which is described as follows:

$$E(S_1, S_2) = \begin{cases} 1 & \text{if } f(S_1) < f(S_2) \text{ or } \frac{A(S_1) - A(S_2)}{A(S_2)} > \frac{f(S_1) - f(S_2)}{f(S_2)} \\ 0 & \text{else.} \end{cases} \quad (6)$$

$E(S_1, S_2) = 1$ implies if the fitness of solution S_1 is less than that of S_2 or the growth rate of acceptability between solutions S_1 and S_2 is bigger than the reduction rate of fitness between these two solutions, which implies that S_1 is a more potential candidate, and 0, vice versa. This is reasonable in that the individual with a higher acceptability (with many better basic elements) is more likely a potential solution. Meanwhile, the new acceptance criterion helps guide individuals to move towards the common ground of all individuals, thus speeding up the convergence of the algorithm and providing a completely new way for an individual to learn from others. The pseudocode of the improved ILS (*i*ILS) is presented in Algorithm 1.

There are many choices for local search. In this paper, we use a fast local search method [17] which calculates the difference in the objective values (i.e., $\Delta f = f(S) - f(S')$) between

```

Initialize population
for i = 1 to n
  Si = LocalSearch(Si)
  fibest = f(Si), Sibest = Si
end for
repeat
  Cal_acceptability();
  for i = 1 to n
    S' = Encapsulate(Si);
    iILS(S', Sibest);
    Si = Unclose(S');
  end for
until termination rule is met

```

ALGORITHM 2: Pseudocode for the PREA.

the current solution S and its neighbor S' . The complexity of the calculation of Δf is reduced by introducing the concepts of “preload” and “postload.” The four operators used in the local search algorithm are insertion, 1-1-interchange, 2-opt, and 2-opt*.

Given the definitions of the preceding section, the PREA can be stated as follows. At first, an initial population is created randomly. Then the processes of ILS, acceptability calculation, and encapsulation are carried out alternately until the termination rule is met. The pseudocode for PREA is shown in Algorithm 2.

In Algorithm 2, n denotes the population size; S_i^{best} and f_i^{best} , respectively, represent the best solution individual S_i has achieved and the corresponding fitness. Cal_acceptability() calculates the basic elements' acceptability on the basis of the available best individual set $\{S_1^{\text{best}}, \dots, S_i^{\text{best}}, \dots, S_n^{\text{best}}\}$. Encapsulate(S_i) is the encapsulation process for the individual S_i , where the selected basic elements of S_i are encapsulated to form a solution for a reduced-size problem. It is used as the starting point for the improved ILS algorithm which aims to find a “better” local optimum of the encapsulated S_i . Unclose(S') transforms the resulting encapsulated solution S' back into the original one.

The general idea underlying the PREA is that the basic elements with higher acceptability are encapsulated and keep constant in the later searching process. With the evolution of the individual solutions and the corresponding reduced problems, the common basic elements of individual solutions are reserved from iteration to iteration. Other basic elements are still passed to the next iterations and evolve in successive searching processes, which helps to keep the diversity of the population. With the increase of common basic elements, all individual solutions will eventually be the same which happens to be the near-optimal solution for the problem.

4. Experimental Results and Analysis

Experiments have been conducted to analyze the performance of the proposed PREA for the CVRP. They were carried out with the standard benchmarks of Christofides

FIGURE 3: Results achieved by PREA with different α and $P_{\max}$.

and Eilon [20] and Golden et al. [21]. The problem set of Christofides includes 14 problem instances C1–C14 and that of Golden 20 instances G1–G20. It is noted that customers in problems C1–C10 are randomly distributed, while in problems C11–C14 customers are distributed into clusters. The problems C1–C5, C11–C12, and G9–G20 are CVRPs, and problems C6–C10, C13–C14, and G1–G8 belong to CDVRPs. For all problem instances, the number of customers ranges from 50 to 483. The proposed PREA is implemented in C++ and runs on an Intel Core E7500 2.93 GHz PC.

The critical point in PREA is how to encapsulate parts of a solution to form a new smaller-size optimization problem. This is achieved by the appropriate choice of parameters α and $P_{\max}$ which are directly related to both the accuracy and the convergence of the PREA. To evaluate the influence of these two parameters, α and $P_{\max}$ are set to different values in our experiments. The test instances are the 5 CVRPs (C1–C5, the number of customers ranges from 50 to 199) of Christofides. For each pair of α and $P_{\max}$, 50 independent runs are carried out for each instance. The results are shown in Figure 3.

In Figure 3, time represents the average running time for a single run while gap (%) evaluates the difference (in percentage) between the average solutions (the average of 50 runs for each instance) and the given best known solutions (BKSs). It is clear that, with the decrease of α and $P_{\max}$, more accuracy solutions can be achieved; yet this consumes more running time. This coincides with the analysis that smaller α and $P_{\max}$ lead to less reduction in size for the reduced problem which implies a larger searching area with more time needed. To evaluate the influence of population size, experiments are also conducted on 5 CVRPs (C1–C5) of Christofides with different n . The parameters α and $P_{\max}$ are set to 0.5 and 0.9, respectively, in the experiments. The results are shown in Figure 4.

Figure 4 shows that by increasing the population size, we can get better results, but this requires more computational time. More individuals in the population make the poor basic elements have less chance to become the common parts of the individuals; thus the solution is improved. Even if n is set to 2, the PREA can find the solutions close to the BSKs in short time. When increasing n to 10, the average results are very close to the BSKs, and the computation time is quite acceptable. By carefully adjusting the three parameters α , $P_{\max}$, and n , the PREA can achieve better solutions quickly. This provides a convenient way for users to compromise between computational time and the quality of solution.

To investigate the convergence performance of the PREA, especially the mechanism of the encapsulation strategy, we illustrate the detailed evolution process for PREA for problem case G10 (323 customers) in Figure 5, with the parameters chosen as $\alpha = 0.5$, $P_{\max} = 0.9$, and $n = 10$. Figure 5(a) displays the evolution of the average scale of the reduced problem in the group. The size of the reduced problem decreases from the original 268 to the final 52, with a drastic decrease happening from around the 180 iterations (we call this acute phase). In no more than 20 iterations, the scale of the problems decreases to one half, while at the beginning this should require more than 100 iterations (we call it obtuse phase). It is worth noting that the gentle decrease of size in obtuse phase is necessary which guarantees a more sufficient and diverse search for each reduced problem to find “better” components to be encapsulated. Of course, the running time complies with the scale of the reduced problem, also decreasing quickly in the acute phase. A careful examination of Figure 5(b) reveals that, in the acute phase, the decrease in fitness is significant and the PREA can still find better solutions efficiently. This is easily explained, since during the later iterations, more and more common basic elements are picked up and encapsulated so

FIGURE 4: Results achieved by PREA with different pop size.

FIGURE 5: The evolution of PREA for case P10.

that PREA can concentrate on smaller and more promising areas, which greatly improves the searching efficiency.

The results for the 14 CVRP instances of Christofides and Eilon and the 20 CVRP instances of Golden et al. are presented in Tables 1 and 2, respectively. The parameters used in the experiments are the same as before. The PREA terminates when all solutions have achieved the same optimum or the maximum iteration 2000 is achieved. Meanwhile, the results achieved by IVND [12] and the recently proposed metaheuristics ILS-RVND-SP [13] are listed for comparison. Similar to PREA, both use ILS as the basic optimizer. To make it easier for comparison, the performance indices considered are the same as those of the counterpart algorithms and

the results are obtained from 10 individual runs. In Tables 1 and 2, the previous best known solutions (the total travel distance/the number of routes) [13, 22] are listed in second column; 3 performance indices are recorded in the following columns: the gap between the best solutions and the BSKs, the average deviation to the BSKs, and also the average time in seconds in a single run. The last three rows give the overall average deviation, average CPU time per problem in seconds, and the computational platform used in different experiments.

The problem sets Christofides and Golden consist of instances which are representative of a variety of CVRP instances including customer variation in scale, distribution,

TABLE 1: Solutions for the CVRPs of Christofides and Eilon [20].

Prob.	BKS	IVND ¹			ILS-RVND-SP			PREA ($n = 10$)			PREA ($n = 20$)		
		Best gap (%)	CPU (s)	Best gap (%)	Best gap (%)	Avg. gap (%)	CPU (s)	Best gap (%)	Avg. gap (%)	CPU (s)	Best gap (%)	Avg. gap (%)	CPU (s)
C1 (50)	524.61/5	0	—	0.00	0.00	0.00	1.5	0.00	0.00	2.6	0.00	0.00	7.7
C2 (75)	835.26/10	0	—	0.00	0.00	0.00	13.5	0.00	0.07	4.9	0.00	0.00	18.2
C3 (100)	826.14/8	0	—	0.00	0.00	0.00	12.5	0.00	0.28	1.6	0.00	0.17	5.3
C4 (150)	1028.42/12	0	—	0.00	0.00	0.03	53.5	0.03	0.37	5.9	0.12	0.35	15.6
C5 (199)	1291.29/16	0.47	—	0.01	0.01	0.15	625.2	0.48	1.16	40.2	0.02	0.75	124.2
C6 (50)	555.43/6	0	—	0.00	0.00	0.19	0.9	0.00	0.29	0.8	0.00	0.09	6.2
C7 (75)	909.68/11	0	—	0.00	0.00	0.04	5.1	0.00	0.21	2.5	0.00	0.15	6.5
C8 (100)	865.94/9	0	—	0.00	0.00	0.00	76	0.11	0.14	1.8	0.00	0.04	6.4
C9 (150)	1162.55/14	0.41	—	0.00	0.00	0.13	82.5	0.00	0.27	12.7	0.00	0.12	74.7
C10 (199)	1395.85/18	0.75	—	0.00	0.00	0.44	496.1	0.39	0.61	35.0	0.35	0.54	101.1
C11 (120)	1042.11/7	0	—	0.00	0.00	0.00	20.7	0.00	1.76	6.1	0.00	0.32	15.5
C12 (100)	819.56/10	0	—	0.00	0.00	0.00	5.2	0.00	0.00	0.4	0.00	0.00	0.8
C13 (120)	1541.14/11	0.12	—	0.00	0.00	0.19	80.2	0.38	1.17	5.5	0.17	0.93	28.9
C14 (100)	866.37/11	0	—	0.00	0.00	0.00	7.1	0.00	0.01	0.7	0.00	0.00	2.4
Avg. gap		0.13		0.00	0.00	0.08		0.10	0.45		0.05	0.25	
Avg. CPU (s)			65.4				100.83			8.82			29.23
Computer		Pentium IV 2.93 G			Intel Core i7 2.93 GHz			Intel Core E7500 2.93 GHz					

¹The average running time and average solution for each problem are not provided.

TABLE 2: Solutions for the CVRPs of Golden et al. [21].

Problem data			IVND ¹			ILS-RVND-SP			PREA (<i>n</i> = 10)			PREA (<i>n</i> = 20)		
Prob.	BKS	Best gap (%)	CPU (s)	Best gap (%)	Avg. gap (%)	CPU (s)	Best gap (%)	Avg. gap (%)	CPU (s)	Best gap (%)	Avg. gap (%)	CPU (s)	Best gap (%)	Avg. gap (%)
G1 (240)	5623.47/9	0.64	—	0.61	0.86	994.6	0.26	1.35	74.6	0.23	0.32	410.29		
G2 (320)	8404.61/10	0.72	—	0.52	0.54	2659.7	0.52	0.61	185.5	0.52	0.52	613.93		
G3 (400)	11036.22/10	0.21	—	0.00	0.00	6065.0	0.00	0.02	131.6	0.00	0.00	380.02		
G4 (480)	13592.88/10	0.23	—	0.23	0.23	10644.5	0.23	0.24	162.0	0.23	0.23	510.43		
G5 (200)	6460.98/5	0.00	—	0.00	0.00	978.7	0.00	0.00	6.5	0.00	0.00	17.08		
G6 (280)	8404.26/7	0.10	—	0.10	0.10	2455.6	0.10	0.11	18.5	0.10	0.10	68.10		
G7 (360)	10102.7/9	1.63	—	0.92	0.92	4410.8	0.92	0.92	121.5	0.92	0.92	237.79		
G8 (440)	11635.3/10	1.22	—	0.65	1.20	7541.8	1.59	1.71	433.6	1.27	1.45	1450.41		
G9 (255)	579.71/14	0.95	—	0.61	0.95	1720.8	0.56	0.80	92.8	0.26	0.58	284.01		
G10 (323)	736.26/16	1.26	—	0.77	1.07	3229.4	0.28	0.64	251.3	0.32	0.44	531.94		
G11 (399)	912.84/18	1.13	—	0.94	1.11	5979.0	0.22	0.33	874.0	0.13	0.22	1865.32		
G12 (483)	1102.69/19	1.38	—	0.96	1.25	10410.7	0.19	0.37	1901.6	0.15	0.27	2896.87		
G13 (252)	85719/26	0.52	—	0.00	0.33	910.4	0.33	0.48	176.7	0.00	0.10	622.49		
G14 (320)	1080.55/30	0.77	—	0.00	0.15	1513.3	0.00	0.24	228.0	0.00	0.03	1363.32		
G15 (396)	133792/33	1.38	—	0.69	0.85	3265.7	0.29	0.42	1072.5	0.07	0.27	4742.69		
G16 (480)	1612.5/37	1.53	—	0.75	0.95	4835.1	0.45	0.60	1746.0	−0.03	0.29	4441.63		
G17 (240)	70776/22	0.16	—	0.00	0.01	937.6	<0.01	0.08	142.5	<0.01	0.02	602.65		
G18 (300)	995.13/27	1.55	—	0.05	0.27	2297.6	1.15	1.30	250.4	0.29	0.79	1430.09		
G19 (360)	1365.6/33	1.25	—	0.05	0.12	2917.3	0.09	0.19	755.7	0.03	0.10	2418.39		
G20 (420)	1818.32/38	1.19	—	0.16	0.29	4997.3	0.14	0.23	2758.6	0.01	0.11	6290.95		
Avg. gap		0.89		0.40	0.56		0.37	0.53		0.23	0.34			
Avg. CPU (s)			1706.40			3938.23			569.19			1558.92		
Computer		Pentium IV 2.93 GHz		Intel Core i7 2.93 GHz			Intel Core E7500 2.93 GHz							

¹The average running time and average solution for each problem are not provided.

FIGURE 6: Solution comparison for M-n200-k16.

FIGURE 7: Solution comparison for G-n262-25.

and possibly additional constraint on traveling distance. Table 1 presents the results for the small-scale Christofides set which vary from case to case with an obvious decrease in computational time yet slightly degraded performance. For the Golden set, the results are more interesting and worth noting. The second last row in Table 2 indicates that for PREA with 10 agents, the performance in best/average gaps is very close to that of the recently proposed ILS-RVND-SP method, but the time needed is only about one-seventh of the latter. This implies that the proposed new algorithm is more efficient for large-scale problems, in the sense that the PREA achieves the comparable best/average gap in shorter time compared with the INVD and ILS-RVND-SP methods. More agents in PREA will result in “better” performance with the cost of requiring more time; with 20 agents, the PREA

has achieved lower best/average gap than ILS-RVND-SP with less running time. To further show the effectiveness of the proposed PREA, we also present 5 new best solutions obtained by PREA during all our experiments, and the new best solutions found for two other examples which are better than the previous best known solutions [23]. These are displayed in Table 3 and Figures 6 and 7.

5. Conclusion

This paper introduces a new population-based evolutionary algorithm called the problem-reduction evolutionary algorithm. The PREA consists of two alternating phases: the searching phase and the evaluation phase. The acceptability of basic elements is obtained in the evaluation phase which

TABLE 3: The new best known solution found by PREA.

Prob.	BKS	New best sol.
G11 (399)	912.84/18	912.33/18
G12 (483)	1102.69/19	1101.83/19
G15 (396)	1337.92/33	1337.87/33
G16 (480)	1612.5/37	1612.01/37
G20 (420)	1818.32/38	1817.86/38

is used to guide the encapsulation in the searching process. Encapsulation is an indispensable scheme for cutting down the scale of the optimization problem. With the evolution of the algorithm, the common basic elements increase to eventually direct to the same optimal. The proposed new algorithm is applied to two benchmark capacitated vehicle routing problems (CVRP) with several hundreds of customers. Experimental results show that the proposed algorithm is superior to the recently proposed comparative algorithms in both convergence speed and solution quality.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grants nos. 61171124, 61373158, and 61301298) and Shenzhen Key Project for Foundation Research (JC201105170613A).

References

- [1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.
- [2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks. Part 1 (of 6)*, vol. 4, pp. 1942–1948, December 1995.
- [3] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization," *Engineering Optimization*, vol. 38, no. 2, pp. 129–154, 2006.
- [4] P. Moscato and M. G. Norman, "A 'memetic' approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," in *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, vol. 1, pp. 177–186.
- [5] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [6] M. Dorigo, V. Maniezzo, and A. Coloni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.
- [7] S. Boettcher and A. G. Percus, "Extremal optimization: methods derived from co-evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 825–832, Morgan Kaufmann, San Francisco, Calif, USA, 1999.
- [8] J. Schneider, "Searching for backbones—a high-performance parallel algorithm for solving combinatorial optimization problems," *Future Generation Computer Systems*, vol. 19, no. 1, pp. 121–131, 2003.
- [9] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, pp. 80–91, 1959.
- [10] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *European Journal of Operational Research*, vol. 218, no. 1, pp. 1–6, 2012.
- [11] F. Li, B. Golden, and E. Wasil, "Very large-scale vehicle routing: new test problems, algorithms, and results," *Computers and Operations Research*, vol. 32, no. 5, pp. 1165–1179, 2005.
- [12] P. Chen, H.-K. Huang, and X.-Y. Dong, "Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1620–1627, 2010.
- [13] A. Subramanian, E. Uchoa, and L. S. Ochi, "A hybrid algorithm for a class of vehicle routing problems," *Computers & Operations Research*, vol. 40, no. 10, pp. 2519–2531, 2013.
- [14] J.-P. Luo, X. Li, and M.-R. Chen, "Improved shuffled frog leaping algorithm for solving CVRP," *Journal of Electronics & Information Technology*, vol. 33, no. 2, pp. 429–434, 2011.
- [15] Y. Nagata and O. Bräysy, "Edge assembly-based memetic algorithm for the capacitated vehicle routing problem," *Networks*, vol. 54, no. 4, pp. 205–215, 2009.
- [16] E. E. Zachariadis and C. T. Kiranoudis, "A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem," *Computers & Operations Research*, vol. 37, no. 12, pp. 2089–2105, 2010.
- [17] W. F. Liu and X. Li, "A fast iterated local searching algorithm for capacitated vehicle routing problem," *Science Paper Online*, 2012.
- [18] W. Liu, X. Li, N. Luo, and X. Chen, "Common grounding optimization for CVRP," in *Proceedings of the IEEE 8th Conference on Industrial Electronics and Applications (ICIEA '13)*, pp. 1755–1758, June 2013.
- [19] E. B. Baum, "Towards practical 'neural' computation for combinatorial optimization problems," in *Neural Networks for Computing*, vol. 151 of *AIP Conference Proceedings*, pp. 53–58, Snowbird, Utah, USA, 1986.
- [20] N. Christofides and S. Eilon, "An algorithm for the vehicle-dispatching problem," *OR*, vol. 20, pp. 309–318, 1969.
- [21] B. L. Golden, E. A. Wasil, J. P. Kelly, and I. M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results," in *Fleet Management and Logistics*, T. G. Crainic and G. Laporte, Eds., Centre for Research on Transportation, pp. 33–56, Springer, New York, NY, USA, 1998.
- [22] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.
- [23] S. Wink, T. Bäck, and M. Emmerich, "A meta-genetic algorithm for solving the capacitated vehicle routing problem," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '12)*, pp. 1–8, Brisbane, Australia, June 2012.