A Set-Based Discrete Differential Evolution Algorithm

Yu Liu, Wei-Neng Chen (corresponding author), Zhi-Hui Zhan, Ying Lin, Yue-Jiao Gong, and Jun Zhang
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Machine Intelligence and Sensor Network, Ministry of Education
Key Laboratory of Software Technology, Education Department of Guangdong Province, P.R. China
junzhang@ieee.org

***Abstract*—The TSP problem is considered as classical discrete optimization grouping problem, which is widely used in practice, but it is real a difficult NP problem. Simultaneously differential evolution (DE) algorithm has been proven to be a powerful optimization algorithm. Since the mutation process of DE contains a series of arithmetic operators operating on continuous space, few algorithms based on DE solve this problem nicely and the advantages of DE in continuous space cannot be used to solve TSP. To take full advantages of the strengths of DE, this paper proposes a set-based DE (S-DE) which completely follows the procedure of the original DE. We present a representation scheme to characterize the discrete problem space and by redefining its basic concept and all related operators in mutation, DE can operate directly on the original set space of the discrete optimization problems instead of performing a space transformation. In that way, the searching features of DE in continuous space is kept. In experiment, we test the performance of our proposed S-DE and the results show it is very promising.**

Keywords-component; Discrete; Differential algorithm; Set-based; COP; TSP

I. INTRODUCTION

Differential Evolution (DE) turns out to be a simple yet powerful algorithm for real parameter optimization since it was proposed by Storn and Price [1] in 1995. DE is a kind of evolutionary algorithms(EA). Similar to the other EAs, it is a population-based stochastic search technique using mutation, crossover, and selection operators at each generation to search the optimum of the problem. By employing this simple and efficient mutation scheme, DE has shown very promising performance in numerical optimizations [2-3] and obtained a series of favorable achievements in the real parameter optimization competitions [4-5]. Due to the success of DE in continuous space, increasingly researchers have been attracted to extend DE to discrete space.

Since many operators in mutation are defined in continuous space, the application of DE in the domain of combinatorial optimization problems (COP) in discrete space is difficult and unusual. Over the past decade, a variety of attempts have been made to apply DE algorithms to solve the discrete optimization problems. In general, these methods can be classified into mainly three categories. The first type is to perform a space transformation based on the continuous relaxation approach [6-8]. These discrete DE algorithms still optimize in the continuous space, but the solutions obtained are converted to discrete ones before the objective function evaluations. The binary relaxation approach [6], rank-based relaxation approach [7], and round down relaxation approach [8] are presented respectively to transform a continuous solution into a discrete one. These relaxation approaches are reasonable if the discrete variables are numerical or a certain order among the variables may be established. But regarding the problems involving the categorical discrete variables, no implicit order exists among the variables, those methods are no longer effective. The second type is to define each individual as a permutation of numbers [9-12]. The traditional discrete DEs based on permutation vector introduce a swap operator [9-11] or a construction and destruction procedure [12] into the mutation schemes. However, these methods deviate from the basic idea of the original DE's mutation schemes that add the differential vector to the base vector. The third type of discrete DE is incorporated with some problem-dependent local search methods [13-16]. This type of algorithms is also called hybrid discrete DE algorithms.

Although a variety of discrete DE algorithms have been proposed, most of the approaches applying the DE to discrete domain took inspirations from the mutation and crossover of DE, but did not follow it exactly. So the advantages of DE in searching the optimum solution in continuous space cannot be fully extended to discrete space. Meanwhile, the hybrid discrete DE algorithms are usually difficult to be extended to solve other COPs because they are designed for specific problems.

In order to extend DE to the discrete space for COPs and keep the searching features of DE in continuous space, a set-based DE is proposed in this paper. The inspiration comes from the set-based PSO [17]. As the same as PSO, DE is also a kind of population-based stochastic optimization algorithm and it also belongs to Evolutionary Algorithm(EA). More importantly, DE shows better performance than PSO for real parameter optimization and the set-based method never change the searching features of algorithms in continuous space, so it is reasonable that we use the same method to extend DE to discrete space. In set-based DE, we define the domain of COPs in the universal set, every candidate solution corresponds to a crisp subset of the universal set. During every iteration, the individual sets and the trial sets are still the feasible solutions while the mutant sets are intermediate variables instead of feasible solutions. All related operators in mutation are

978-1-4799-0652-9/13 $31.00 © 2013 IEEE
DOI 10.1109/SMC.2013.233

redefined on crisp sets to enable S-DE to follow the structure of the original DE which brings a huge benefit that our proposed S-DE can keep the searching feature of the original DE in continuous space. In the experiment, we compared our proposed S-DE with ACS[24], MMAS [25], S-CLPSO [17] and some improved DE in TSP. The result shows our proposed S-DE is very promising.

The rest of this paper is organized as follows. In Section 2, a review on the original DE is made. In Section 3, the set-based DE is proposed. The performance of S-DE on TSP is shown in Section 4. Conclusion is drawn in Section 5.

II. DE In Continuous Space

Differential Evolution (DE) is a population-based stochastic algorithm which utilizes *NP* *D*-dimensional parameter vectors as a population to search for the global numerical optimum. The DE Algorithm starts with initializing the target population $X=[x_i^j]_{NP\times D}$ with the size of *NP* and the dimension of *D*. *NP* and *D* does not change during the whole process. The *i*th(*i*=1,...,*NP*) *D*-dimensional parameter vector can be described as $x_i=[x_i^1,x_i^2,...,x_i^D]$ which is initialized as a random point in the solution space. Then the algorithm works through a simple cycle of stages, i.e., mutation, crossover and selection to find the global optimum. We will explain each stage separately.

A. Mutation

For each target vector $x_i=[x_i^1,x_i^2,...,x_i^D]$ (*i*=1,..., *NP*), a mutant vector is generated according to

$$v_i = x_{r_1} + F\cdot(x_{r_2}-x_{r_3}) \tag{1}$$

With random indexes $r_1,r_2,r_3\in\{1,2,...,NP\}$ which are different from each other and the running index *i*, so that *NP* must be greater or equal to four to allow for this condition. The scale number *F* is a positive control parameter used to scale the differential vector $(x_{r_2}-x_{r_3})$. *D* is the dimension of the search space.

Price and Storn proposed several strategies of mutation [8] based on the number of differential vectors and the individuals being perturbed. The strategy talked above is denoted as DE/rand/1, meaning that the vector to be perturbed is selected randomly and only one differential vector is used. There are also other strategies of mutation as it is shown in [18], and in most cases the DE/rand/1 strategy shows better performance [19], so in this paper we use DE/rand/1 strategy to test TSP.

B. Crossover

The crossover operation is used to enhance the diversity of the population. To this end ,the trial vector:

$$u_i=(u_i^1,u_i^2,...,u_i^D),\ i=1,2,...,NP \tag{2}$$

is formed according to

$$u_i^j=\begin{cases} v_i^j & \text{if } \mathrm{rand}(0,1)<CR \text{ or } j=j_{\mathrm{rand}} \\ x_i^j & \text{otherwise}\end{cases},\ j=1,2,...,D \tag{3}$$

In (3), rand(0,1) is a random number between 0 and 1, *D* is the dimension of the search space, j_{rand} is a randomly chosen index which ensures u_i gets at least one parameter from v_i.

This kind of crossover is called binomial crossover. There is another crossover called exponential crossover [18]. Two variants *st* and *L* is used in this kind of crossover. Variant *st* denotes the start point from where the trial vector inherits from the mutant vector. *L* denotes the number of components the trial vector inherits from the mutant vector. The number *st* is generated randomly from $\{1,2...D\}$ and *L* is formed according to the following pseudo-code.

```
For ($L$ = 1; rand(0,1) < $CR$ && $L < D$; $L$ ++);
```

After *st* and *L* are generated, the trial vector is obtained as

$$u_i^j=\begin{cases} v_i^j & \text{if } j=\{st \bmod D,(st+1) \bmod D, \\ & ...,(st+L-1) \bmod D\} \\ x_i^j & \text{otherwise}\end{cases},\ j=1,2,...,D \tag{4}$$

C. Selection

The one with better fitness between the target vector and the trial vector will be chosen to form the next generation. For a minimization problem, the vector with a lower fitness value survives to the next generation, which is expressed as follows.

$$x_i=\begin{cases} u_i & \text{if } f(x_i) \text{ is better than } f(u_i) \\ x_i & \text{otherwise}\end{cases},\ i=1,2,...,NP \tag{5}$$

where *f*(*x*) is the objective function.

III. The Set-based DE

In this section, the set-based DE method is described. Our purpose when proposing the S-DE is to extend the original DE to solve the discrete space optimization problems without changing its procedure so that the S-DE can extend the searching features of the original DE in continuous space to the discrete space. To allow this, S-DE applies a set-based representation scheme and redefines all related operators for the discrete space. We can see its structure in Figure .1.

```
procedure S-DE
    initialization;
    while terminal condition not met
        for each individual set $i$($i$=1,2,...,$NP$)
            mutation;
            crossover;
            selection;
        end for
    end while
end procedure
```

Figure 1. The structure of S-DE

A. Representation Scheme

According to [20], many COPs can be formulated in the abstract as "find from a set E a subset that satisfies some constraints and optimizes the objective function." Following this scheme, in S-DE, a COP is described by the following characteristics.

- The domain of a COP is defined in the universal set *E*, which can be divided into *n*-tuple $(E^1, E^2, ..., E^n)$, where *n* can be regarded as the dimension of the searching space and E^i (i=1,2,...n) is a set corresponding to the *i*th dimension of the searching space. For example, in TSP, the universal set *E* is composed of all arcs connecting every two cities and the *j*th (j=1,2,...,n) dimension of the searching space E^j is the set of arcs which are connected with node *j*.
- A solution to the problem *X* is also an *n*-tuple $(X^1, X^2, ..., X^n)$, where X^i (i=1,2,...n) is a crisp set and $X^i \in E^i$. *X* is feasible only if *X* satisfies the constraints Ω. In symmetric TSP, Any feasible solution *X* form a Hamiltonian circuit of the graph. *X* can be described as a set and be divided into *n*-tuple $(X^1, X^2, ..., X^n)$ and X^i (i=1,2,...n) is a subset of E^i with two arcs. This is because there must be two arcs connected with node *i*.

After giving the scheme, our goal of solving the COPs in the discrete space is to find a feasible solution that optimizes the objective function. Figure. 2 gives us an example of the symmetric TSP to help us understand this representation.

B. Procedure of the Algorithm

In our S-DE, following the original DE, we first initialize *NP* random feasible solutions as a population to search for the global numerical optimum. Then we start searching the optimization iteratively. During every iteration, we also have three stages , i.e., mutation, crossover and selection. Since the operators of the original DE can only be used in continuous space, we redefine all related operators for the discrete space which we will discuss below in detail.

1) Mutation

In S-DE, the mutation operator generates *NP* mutant sets, *NP* is the size of the population. The *i*th mutant set is generated according to

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \tag{6}$$

All parameters has the same meaning as described in (1). We

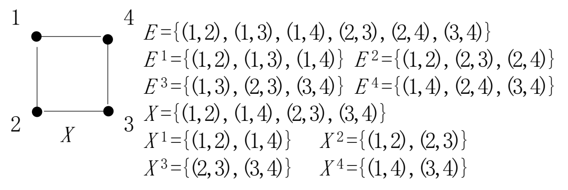

1
4
2
3
X
E={(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)}
E^1={(1,2), (1,3), (1,4)} E^2={(1,2), (2,3), (2,4)}
E^3={(1,3), (2,3), (3,4)} E^4={(1,4), (2,4), (3,4)}
X={(1,2), (1,4), (2,3), (3,4)}
X^1={(1,2), (1,4)} X^2={(1,2), (2,3)}
X^3={(2,3), (3,4)} X^4={(1,4), (3,4)}

Figure.2 Example for the symmetric TSP

redefines some operators in (6) as follows.

Operator 1) Individual set − Individual set → Differential set

The minus operator between two crisp set A and B is defined as follows:

$$A - B = \{e \in A \text{ and } e \notin B\} \tag{7}$$

We apply this operator definition to our S-DE. For example, A={(1,2),(2,3),(3,4),(1,4)} and B={(1,3),(3,2),(2,4),(4,1)}, then following Operator 1, $A-B$={(1,2),(3,4)}.

Operator 2) Scale factor × Differential set → Scaled differential set

We define this operator as that every dimension of differential set survives with the probability *F*. *F* is the scale factor. Here in our algorithm , $F \in [0,1]$. We describe this as follows:

$$F \times d^j = \begin{cases} d^j & \text{if rand}(0,1) < F \\ \varnothing & \text{otherwise} \end{cases} \quad ,j=1,2,...,n \tag{8}$$

which d^j denotes the *j*th dimension of the differential set and *n* denotes the dimension of the search space. For example, $d^2 = \{(1,2),(2,4)\}$, F is set as 0.8, r=rand(0,1)=0.4, so $F \times d^2$ = d^2 ={(1,2),(2,4)}. But in the same case, if r=rand(0,1)=0.9, $F \times d^2 = \varnothing$.

Operator 3) Scaled differential set + Scaled differential set → Scaled differential set

The plus operator between two differential set A and B is defined as follows:

$$A + B = \{e \in A \text{ or } e \in B\} \tag{9}$$

For example, A={(1,2),(3,4)}, B={(1,2),(2,3)}. Then A+B={(1,2),(2,3),(3,4)}.

Operator 4) Individual set + Scaled differential set → Mutant set

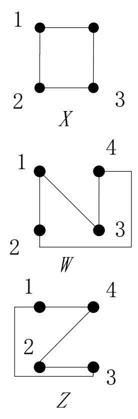
We define this operator as that the individual set is replaced by the scaled differential set to cause a deviation in the corresponding dimension if the latter is not null.

$$x^j + SD^j = \begin{cases} x^j & \text{if } SD^j = \varnothing \\ SD^j & \text{otherwise} \end{cases} \quad ,j=1,2,...,n \tag{10}$$

For example, suppose x^3 ={(1,3),(3,4)}, if SD^3 ={(2,3)}, x^3 + SD^3 ={(2,3)}. But if SD^3 = $\varnothing$, the result will be {(1,3),(3,4)}.

Based on the four operators, the mutation stage can be extended to discrete space. After that the mutant set is no long a feasible solution of the problem, but we can still use it as intermediate variable to generate the trial set. Figure.3 helps to understand the procedure of mutation for symmetric TSP.

2) Crossover



X={(1,2),(2,3),(3,4),(1,4)}
W={(1,2),(2,4),(3,4),(1,3)}
Z={(1,4),(2,4),(2,3),(1,3)}
$V=X+F*(W-Z)$

X^1={(1,2),(1,4)} X^2={(1,2),(2,3)}
X^3={(2,3),(3,4)} X^4={(1,4),(3,4)}
$D=W-Z$={(1,2),(3,4)} $SD=F*D$
$D^1=D^2$={(1,2)} $D^3=D^4$={(3,4)}

Suppose that F=0.7 in (6) and four random floating number r1=0.9>F, r2=0.4<F, r3=0.6<F,r4=0.7=F. So SD^1 is null , SD^2= {(1,2)}, SD^3={(3,4)}, SD^4 is null.
$V^1=X^1$={(1,2),(1,4)} $V^2=SD^2$={(1,2)}
$V^3=SD^3$={(3,4)} $V4=X^4$={(1,4),(3,4)}
V={(1,2),(1,4),(3,4)}

Figure.3 The example of mutation for symmetric TSP

Crossover is introduced to increase the diversity of the perturbed parameter vectors. We will take the binomial crossover for example. The exponential crossover is very similar. The whole procedure is shown in Figure. 4 and Figure. 5. The trial set $u = u^1 \cup u^2 \cup \cdots \cup u^D$ is also a feasible solution. D is the dimension of the searching space. u^j (j=1,2,...,D) is the jth dimension of the trial set u. We generate u_i (i=1,2,...,NP) as the following steps. NP is the size of the population.

Step 1) A random index $0<jrand<D$ is generated.

Step 2) For each dimension j(j=1,2,...,D), a random variant $r \in [0,1]$ is generated.

Step 3) If $r<CR$ or $j=jrand$, u_i^j learns from v_i^j .

Step 4) If $r>CR$ and $r \neq jrand$, u_i^j learns from x_i^j .

There is a learn procedure in Step 3 and Step 4. Figure. 5 shows the procedure of which u_i^j learns from a set s_i^j . In the learn procedure, a selection operator is introduced. When there are more than one elements available, we can randomly choose an element which is called randomly element selection operator or choose an element based on some problem-independent information which is called heuristic-based element selection operator.

We will take TSP for example to facilitate understanding. In TSP, the length of each edge can be taken as the heuristic information and the edge with the shortest length is selected. In this case, we assume that the dimension is 4 which means there are 4 cities in this case, CR=0.4, the individual set X is set as {(1,2),(2,3),(3,4),(1,4)} and the mutant set V is set as {(1,3),(1,2),(3,4)}. Our goal is to generate the trial set U. First, a random index $jrand$ is generated. Here we assume $jrand$=2. Then we start with the 1st dimension. A random variant r is generated, assuming r=0.6. Since $r>CR$, U^1 should learns from X^1 . Here we select the arc (1,2) to add to U^1 . Then we turn to the 2nd dimension. This time r=0.8. Since $j=jrand$=2, U^2 learns from V^2 . There is only one arc (1,2) in V^2 which has been selected to add to U^1 . So we select an arc from E^2 which is the set of arcs which are connected with Node 2. Supposing we select (2,4), then we turn to dimension 4. We repeat the procedure to finish the construction of U.

But for TSP, exponential crossover has its own meaning. The effect of the exponential crossover is connecting the continuous path of the mutant set and the individual set, which could be destroyed by the binomial crossover. So in this paper, we use the exponential crossover to test the TSP. In the example above, first, we randomly select a variable named st and a variable named L. We turn to dimension st and in the next L steps, U learns from V. After that U learns from X to finish the construction.

3) *Selection*

The selection operator is similar with the original DE. The one with better fitness between the target set and the trial set will be chosen to form the next generation. For a minimization problem, the set with a lower fitness value survives to the next generation, which is expressed as follows.

$$x_i = \begin{cases} u_i & \text{if } f(x_i) \text{ is better than } f(u_i) \\ x_i & \text{otherwise} \end{cases}, \; i=1,2,..., NP \quad (11)$$

```
procedure crossover( $x_i$ , $v_i$ )
$u_i = \varnothing$ ;
generate a random index jrand;
    for each dim j (j=1,2,...,D)
        generate a random variant $r \in [0,1]$ ;
        If ( r<CR or j=jrand)
            learn( $u_i^j$ , $v_i^j$ );
        end if
        else
            learn( $u_i^j$ , $x_i^j$ );
        end else
    end for
end procedure
```

Figure 4. Example of a figure caption Pseudo-code of crossover procedure.

```
procedure learn( $u_i^j$ , $s_i^j$ )
    mid_set={e|e $\in s_i^j$ and e satisfies $\Omega$ };
    while $u_i^j$ is not finished and mid_set is not empty
        select a element from mid_set and add it to $u_i^j$
        update mid_set;
    end while
    mid_set={e|e $\in E^j$ and e satisfies $\Omega$ };
    while $u_i^j$ is not finished and mid_set is not empty
        select a element from mid_set and add it to $u_i^j$
        update mid_set;
    end while
end procedure
```

Figure 5. Pseudo-code of learn procedure.

where $f(x)$ is the objective function for the minimization problem.

IV. Experimental Results

We use TSP to test our S-DE. The TSP instances are derived from the TSPLIB [23]. S-DE employs the most classical rand/1 mutation scheme which has been proved to exhibit favorable exploration ability [19] and the exponent crossover scheme as discussed in Section 3. The heuristic elements selection operator is employed in the crossover process to select the shortest edges. The parameters CR and F are set as 0.9 and 0.5 respectively as in most literature that employs a exponential crossover [21-22]. The total problem evaluations for all the algorithms are set as $500n$, where n is the number of cities.

A. Comparisons with Discrete DE Algorithms

We first compare the S-DE with several state-of-the-art discrete DE algorithms, i.e., the discrete DE (DDE) proposed in [13], the discrete DE with smallest value position rules (DDE-SVP) mutation strategy [14], and the ensemble discrete DE (eDDE) algorithm [12]. The parameter settings of these discrete DE algorithms for comparison follow the original papers. To make fair comparisons, all these algorithms employ a completely random initialization and do not contain the local search heuristics. The simulation results are exhibited in Table I.

From Table I we can see that S-DE greatly outperforms the compared state-of-the-art discrete DE algorithms in all the 17 TSP instances in both the average and best results. Moreover, the best results of S-DE in the 20 runs can reach the world-known optimal results on 16 of the 17 test instances, whereas the compared discrete DE algorithms failed to obtain the optimal results on any of the instances. Compared with the state-of-the-art discrete DE algorithms, our proposed S-DE algorithm directly optimize in the set space of TSP without a space transformation in advance. Meanwhile, S-DE is very similar with the basic idea of the original DE no matter in the mutation process or in the crossover process. Moreover, our proposed S-DE shows favorable performance on TSPs which greatly overtakes the state-of-the-art discrete DE algorithms.

B. Comparisons with other Algorithms

The S-DE is then compared with some other metaheuristic algorithms, i.e., ACS [24], MMAS [25], and S-CLPSO [17] as illustrated in table II. Among the compared algorithms, ACS and MMAS are predominantly defined for combinatorial optimization problems and exhibit good performance. The S-CLPSO is a recently proposed discrete PSO algorithm which shows favorable results on combinatorial optimization problems. The parameter settings of ACS, MMAS and S-CLPSO follow the original literature [24][25][17]. The results in Table II illustrate that even compared with the state-of-the-art metaheuristic algorithms, S-DE is still competitive. S-DE can obtain the best average results in all the tested TSP instances. In addition, the best results of ACS, MMAS, and S-CLPSO in 20 runs are worse than those of S-DE in most test instances. Meanwhile, we also compare the average results of SDE for the 17 tested instance with the best results of ACS, MMAS and SCLPSO.9 are better than those of ACS, 17 are better than those of MMAS and 7 are better than those of SCLPSO.

TABLE I The Performance of S-DE Compared with the State-of-the-art Discrete DE Algorithms on TSP

Instance name	Best known results	S-DE		DDE		DDE-SVP		Ensemble DDE	
		Best	Ave	Best	Ave	Best	Ave	Best	Ave
eil76	538	**538**	**538**	1459	1540.4	1007	1601.4	555	591.9
eil51	426	**426**	**426.85**	895	940.25	676	864.4	445	455.7
eil101	629	**629**	**629.3**	2016	2189.9	1517	2198.45	682	717.1
st70	675	**675**	**675.55**	1966	2125.65	1225	1976.3	703	764.9
kroa100	21282	**21282**	**21321.55**	96755	101152.3	52364	76323.3	24330	26011.45
kroa150	26524	**26524**	**26539.05**	158426	167187.6	90088	135466.1	30714	33394.05
krob100	22141	**22141**	**22166.5**	93369	99523.1	52395	69987.45	23963	26239.3
krob150	26130	**26130**	**26141.25**	155330	163639.4	94291	133835.7	30787	33799.35
krob200	29437	**29437**	**29468.4**	218769	227128.9	151360	222173.5	35547	39959.95
lin105	14379	**14379**	**14419.2**	69336	72672.6	36951	48245.15	16136	17466.8
pr107	44303	**44303**	**44411.3**	285500	306046.5	84030	154220.4	49251	52455.85
pr144	58537	**58537**	**58595.7**	490197	523368.9	423767	506934.2	73329	81308.5
pr136	96772	**96785**	**96912.1**	522341	542591	326694	513711.5	111730	122818.3
pr152	73682	**73682**	**73710.3**	654057	681890.8	427803	660290.5	79576	92101.55
pr299	48191	**48191**	**48294.6**	527763	544853.4	392115	583010.2	61985	67981.65
berlin52	7542	**7542**	**7542**	14346	16197.75	11653	16382.35	7988	8312.3
tsp225	3916	**3916**	**3922.15**	28789	29701.15	26757	32815.65	4609	5090.7

TABLE II The Performance of S-DE Compared with other Heuristic Algorithms on TSP

Instance name	Best known results	S-DE		ACS		MMAS		S-CLPSO	
		Best	Ave	Best	Ave	Best	Ave	Best	Ave
eil76	538	**538**	**538**	541	550.05	546	548.45	**538**	539.45
eil51	426	**426**	**426.85**	427	432.05	429	436.6	**426**	427.25
eil101	629	**629**	**629.3**	639	648.6	641	660.25	**629**	638.5
st70	675	**675**	**675.55**	**675**	686.45	680	698.95	**675**	676.45
kroa100	21282	**21282**	**21321.55**	**21282**	21577.55	21594	22059.6	**21282**	21348.65
kroa150	26524	**26524**	**26539.05**	26875	27265.3	26959	27420.75	26663	26933.15
krob100	22141	**22141**	**22166.5**	22216	22447.85	22287	22468.25	**22141**	22227.1
krob150	26130	**26130**	**26141.25**	26345	26763.2	26806	27161.3	26141	26465.45
krob200	29437	**29437**	**29468.4**	29767	30437.85	30443	31564.1	29539	30089.9
lin105	14379	**14379**	**14419.2**	**14379**	14571.65	14461	14594.25	**14379**	14433.15
pr107	44303	**44303**	**44411.3**	**44303**	44573.25	44581	45179.2	**44303**	44683.75
pr144	58537	**58537**	**58595.7**	**58537**	58643.35	58679	58834.8	58657	58771.4
pr136	96772	**96785**	**96912.1**	97293	100351	104300	106434.9	97279	98410.6
pr152	73682	**73682**	**73710.3**	73818	74172.2	74136	74528.85	**73682**	74126.2
pr299	48191	**48191**	**48294.6**	48871	49695.7	51519	53022.2	48466	49159.45
berlin52	7542	**7542**	**7542**	**7542**	7634	7547	7640.45	**7542**	7544.55
tsp225	3916	**3916**	**3922.15**	3965	4060.45	4043	4108.75	3921	3959.3

V. Conclusion

In this paper, we proposed a set-based DE to extend the original DE to discrete space by redefining the basic concepts and operators of the original DE according to the set representation scheme. The mutation and crossover always stay the same with those of the original DE so that the proposed S-DE can totally keep the searching features of DE in continuous space. By comparing the results of TSP implemented by S-DE, ACS, MMAS and S-CLPSO, we can draw conclusion that the performance of S-DE to TSP is very promising. In further research, we will try to develop our S-DE to solve more COPs in S-DE.

Acknowledgment

This work was partially supported by the National High-Technology Research and Development Program (“863” Program) of China under Grand No. 2013AA01A212, by the National Science Fund for Distinguished Young Scholars under Grant 61125205, by the National Natural Science Foundation of China under Grant 61070004 and 61202130, by the NSFC Joint Fund with Guangdong under Key Project U1201258 and U1135005.

References

[1] R. M. Storn and K. V. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," J. Global Optim, vol. 11, 1997, pp. 341–359.

[2] A. K. Qin, V. L. Huang and P. N. Suganthan," Differential evolution algorithm with strategy adaptation for global numerical optimization," IEEE Trans. Evol. Comput., vol. 13, 2009, pp. 398-417.

[3] Y . Wang, Z. Cai, Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," IEEE Trans. Evol. Comput., vol. 15, 2011, pp. 55-66.

[4] R. Storn and K. V. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," In Proceedings of IEEE International Conference on Evolutinary Computation, 1996, pp. 842-844.

[5] K. V. Price, "Differential evolution versus the functions of the 2nd ICEO," In Proceedings of IEEE International Conference on Evolutinary Computation, Apr 1997.

[6] A. P. Engelbrecht and G. Pampara, "Binary Differential Evolution Strategies," In Proceedings of IEEE Conference on Evolutinary Computation, 2007.

[7] J. Zhang, Viswanath Avasarala, Arthur C. Sanderson and Tracy Mulle, "Differential Evolution for Discrete Optimization: An Experimental Study on Combinatorial Auction Problems," In Proceedings of IEEE Conference on Evolutinary Computation, 2008.

[8] N. Damak, B.Jarboui, P. Siarryb and T. Louki, "Differential evolution for solving multi-mode resource-constrained project scheduling problems," Computers & Operations Research, vol. 36, 2009, pp. 2653-2659.

[9] M. F. Tasgetiren, Quan-Ke Pan, P. N. Suganthan and Yun-Chia Liang, "A Discrete Differential Evolution Algorithm for the No-Wait Flowshop Scheduling Problem with Total Flowtime Criterion," In Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, pp. 251-258 SCIS'07, April, 2007.

[10] M. F. Tasgetiren, Q.-K. Pan, Y.-C. Liang and P. N. Suganthan, "A Discrete Differential Evolution Algorithm for the Total Earliness and Tardiness Penalties with a Common Due Date on a Single-Machine," In Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, pp. 271-278, April 2007.

[11] Q.-K. Pan, M. Fatih Tasgetiren and Yun-Chia Liang, "A Discrete Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem," presented at the Genetic Evol. Comput., 2007.

[12] M. F. Tasgetiren, P.N. Suganthan and Quan-Ke Pan, "An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem." Applied Mathematics and Computation, vol. 215, 2010, pp. 3356–3368.

[13] L. Wang, Q.-K. Pan, P.N. Suganthan, Wen-Hong Wang and Ya-Min Wang, "A novel hybrid discrete differential evolution algorithmfor blocking flowshop scheduling problems," Computers & Operations Research, vol. 37, 2010, pp. 509-520.

[14] J. G. Sauer and L. d. S. Coelho, "Discrete Differential Evolution with Local Search to Solve the Traveling Salesman Problem: Fundamentals and Case Studies," In Proceedings of IEEE International Conference on Cybernetic Intelligent Systems, 2008, pp. 1-6.

[15] Q.-K. Pan, L. Wang, L. Gao and W.-D. Li, "An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers," Information Sciences, vol. 181, 2011, pp. 668-685.

[16] G. Deng and X. Gu, "A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion," Computers & Operations Research, vol. 39, 2011, pp. 2152-2160.

[17] W.-N. Chen, Jun Zhang, Henry Chung, Wen-liang Zhong, Wei-Gang Wu and Yu-hui Shi, "A novel set-based particle swarm optimization method for discrete optimization problem," IEEE Transactions on Evolutionary Computation, vol. 14, no. 2, 2010, pp. 278-300.

[18] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," IEEE Trans. Evol. Comput., vol. 15, 2011, pp. 4-31.

[19] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," In Proc. Genet. Evol. Comput. Conf., 2006, pp. 485-492.

[20] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," Operations Res.,vol. 21, no.2, 1973, pp. 498-516.

[21] H. Wang, Z. Wu and S. Rahnamayan, "Enhanced Opposition-Based Differential Evolution for Solving High-Dimensional Continuous Optimization Problems," Soft Computing, vol. 15, 2010, pp. 1-14.

[22] F. Herrera, M. Lozano, and D. Molina, "Components and Parameters of DE, Real-coded CHC, and G-CMAES," University of Granada, Spain2010.

[23] TSPLIB, Http://www.Iwr.uni_heidelberg.de

[24] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation,* vol. 1, pp. 53-66, 1997.

[25] T. Stützle and H. Hoos, "MAX-MIN ant system and local search for the traveling salesman problem," presented at the IEEE International Conference on Evolutionary Computation (ICEC'97), 1997.