Multi-start JADE with knowledge transfer for numerical optimization

Fei Peng, Ke Tang ,Guoliang Chen and Xin Yao

***Abstract*— JADE is a recent variant of Differential Evolution (DE) for numerical optimization, which has been reported to obtain some promising results in experimental study. However, we observed that the reliability, which is an important characteristic of stochastic algorithms, of JADE still needs to be improved. In this paper we apply two strategies together on the original JADE, to dedicatedly improve the reliability of it. We denote the new algorithm as rJADE. In rJADE, we first modify the control parameter adaptation strategy of JADE by adding a weighting strategy. Then, a "restart with knowledge transfer" strategy is applied by utilizing the knowledge obtained from previous failures to guide the subsequent search. Experimental studies show that the proposed rJADE achieved significant improvements on a set of widely used benchmark functions.**

I. Introduction

Differential Evolution (DE) has been shown to be a simple yet powerful evolutionary algorithm for global optimization problems [1], [2]. Recently, a new variant of DE, namely Adaptive Differential Evolution with Optional External Archive (JADE), has been developed for numerical optimization, and showed its superiority over other evolutionary algorithms according to the reported results [3]. However, JADE shows somehow lack of reliability on some problems, i.e., it can find the global optima for some runs, but it also has the probability of getting trapped into local optima.

Reliability is one of the two important characteristics of an evolutionary algorithm, as well as the convergence speed [4]. As discussed in [4], reliability deals with the question of how much chance the algorithm can obtain a "good" solution. To evaluate the reliability of an evolutionary algorithm, several approaches have been proposed in previous literature [5], e.g., one commonly used measure is the success rate (SR) over a number of independent runs [5], [21]. Reliability is especially important for multimodal problems [6]. In the past decades, many approaches have been developed both to increase the reliability and to improve the convergence speed of evolutionary algorithms [7], [8], [9], [10], [11], [12], [13], [14], [22]. However, by observing the results presented in [3], we find that the reliability of the original JADE on some multimodal problems still needs to be improved.

In many real-world problems, the optimal solution is usually not known in prior and the best solution found so far is required to be reported before a given deadline. We call this process an independent run. Under this circumstance, once the chosen algorithm fails in a run, an alternative choice would be to restart the algorithm. However, the algorithm may fall into previous failures again, which will make all the time wasted. In this case, recording the information obtained from previous failures and utilizing the information to guide the subsequent search may benefit all the search process. Based on this idea, we propose two simple yet effective strategies and apply them to the original JADE (JADE with external archive), to form our new algorithm rJADE. In rJADE, two mechanisms are utilized: a modified self-adaptation for controlling crossover rate CR, and the "restart with knowledge transfer" strategy. For the control parameter adaptation, we modify the original control parameter self-adaptation of JADE by adding a weighting strategy. The "restart with knowledge transfer" strategy is applied as follows: whenever the current population converges at a local optimum, a restart is activated. Hence, a run of rJADE may consist of several restarts. We call each restart a phase of the run. The information gained from previous failures will be recorded as knowledge, and will be transferred to the next phase to guide the subsequent search. In our approach, the areas around the local optima found in previous failures are treated as the knowledge. The newly generated population in the new phase will be forced away from previous local optima. Furthermore, to make the population be capable of searching around previous local optima, a perturbation operation is applied whenever the restarted population falls into previous local optima again. Thus, our approach would both have the ability of searching away from previous local optima and around these local optima, and this may increase the chance of finding the global optima finally.

The restart strategy applied in our approach has also been employed in some previous literature. For example, Wang et al. applied a restart strategy in estimation of distributed algorithm [15] and A. Auger et al. proposed a restart CMA Evolution Strategy with increasing population size [16]. However, none of them utilized the knowledge transfer scheme. The knowledge used in our approach is similar as the tabu list used in Tabu Search (TS) [17]. In fact, in previous literature, hybridizing evolutionary algorithms with TS has also been studied. For example, in [18], a hybrid optimization technique combining TS with a hybrid particle swarm optimization and sequential-quadratic-programming technique (hybrid PSO-SQP) has been proposed for unit commitment problems (UC problems). In [18], the TS method is used to solve the combinatorial part of the UC problem while PSO-

The authors are with the Nature Inspired Computation and Applications Laboratory, the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (emails: pzbhlp@mail.ustc.edu.cn, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).

Corresponding author: Ke Tang (Phone: +86-551-3600754).

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

SQP is used to solve the nonlinear-optimization part. Wang et al. in [19] recently proposed another approach (named TPSO) by incorporating tabu list into PSO. In TPSO, the local optima and their neighborhood areas are recorded by a tabu list; the individuals flying into those recorded areas again will be reset. However, none of the two utilize the restart strategy, and thus the search process will not be divided to multiple phases. As a result, no knowledge transfer happens in their approaches.

To demonstrate the efficacy of the proposed rJADE over original JADE, we evaluate both of them on test functions selected from both the classical benchmark functions [20] and CEC2005 benchmark functions [21]. All of these test functions are of minimization problems. Empirical results show that, by applying the proposed strategies on original JADE, rJADE improves the reliability of JADE significantly. Besides, by comparing the performance of rJADE with a restart JADE without knowledge transfer, we can find the superiority of modifying the control parameter adaptation and using the knowledge to guide the search process. Finally, we compare our approach with a state-of-art evolutionary algorithm, namely restart CMA evolution strategy with increasing population size (IPOP-CMA-ES) [16]. We find that rJADE showed better or at least comparable reliability than IPOP-CMA-ES on the test problems, while took longer time to converge.

The rest of the paper is organized as follows: Section II first gives some preliminaries about Differential Evolution and introduces the original JADE, and then presents the proposed rJADE. Experimental study is presented in Section III. Finally, we draw the conclusion and discuss future work in Section IV.

II. Restart JADE with Knowledge Transfer

A. Differential Evolution (DE)

Differential Evolution [1], [2] follows the general framework of an evolutionary algorithm. Individuals in DE at generation g are represented by D-dimensional vectors $\mathbf{x}_{i,g} = (x_{i,1,g}, x_{i,2,g}, ..., x_{i,D,g}), \forall i \in \{1, 2, ..., NP\}$, where D is the dimension of the problem and NP is the population size. After initialization of the population, the classical DE algorithm enters a loop of evolutionary operations: mutation, crossover and selection.

1) Mutation: For each target vector $\mathbf{x}_{i,g}$ at generation g, DE creates a corresponding mutation vector $\mathbf{v}_{i,g}$. Then, after the mutation there will be NP $\mathbf{v}_{i,g}$'s for creating the same number of offspring in generation g. The following three mutation strategies are frequently used in previous literature:

- 'DE/rand/1':

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (1)$$

- 'DE/current-to-best/1':

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{best,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (2)$$

- 'DE/best/1':

$$\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (3)$$

where $r0, r1, r2$ are random and mutually different integers chosen from the set $\{1, 2, ..., NP\} - \{i\}$. $\mathbf{x}_{best,g}$ is the best vector at generation g. In classic DE, scale factor $F > 0$ is a single real constant factor which usually ranges on $(0, 1]$, while in many adaptive DE algorithms each individual i is associated with its own mutation factor F_i, which also ranges on $(0, 1]$.

2) Crossover: After mutation, a 'binary' crossover is usually used to form the final trial vector $\mathbf{u}_{i,g} = (u_{i,1,g}, u_{i,2,g}, ..., u_{i,D,g})$:

$$u_{i,j,g} = \begin{cases} v_{i,j,g} & \text{if } rand_j(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j,g} & \text{otherwise} \end{cases} \quad (4)$$

where $rand_j(l, u)$ is a uniform random number between l and u, and j_{rand} is a randomly chosen index within $[1, D]$ to ensure that the final trial vector $\mathbf{u}_i$ does not duplicate the base vector $\mathbf{x}_i$. In classic DE, crossover rate $CR \in [0, 1]$ is a single parameter that is used to generate all trial vectors, while in many adaptive DE's, each individual i is associated with its own crossover rate CR_i which ranges on $[0, 1]$.

3) Selection: Without loss of generality, we consider only minimization problems here. Then, the selected vector is generalized by

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases} \quad (5)$$

A selection is called a *successful update* if the fitness value of the trial vector $\mathbf{u}_{i,g}$ is better than the fitness value of the parent vector $\mathbf{x}_{i,g}$. Accordingly, the control parameters F_i and CR_i used to generate $\mathbf{u}_{i,g}$ which obtains a *successful update* are called *successful mutation factor* and *successful crossover rate*, respectively.

B. Introduction to JADE

JADE (Adaptive Differential Evolution with Optional External Archive) [3] is recently proposed by J. Zhang and A. Sanderson to improve convergence performance, by implementing a new mutation strategy 'DE/current-to-p-pest with/without external archive' and adaptively controlling the parameters. JADE adopts the same 'binary' crossover and one-to-one selection scheme as many other classic DE variants do as described in Eq. (4) and Eq. (5). Main contributions of JADE are summarized as follows:

- 'DE/current-to-p-best with/without external archive': 'DE/current-to-p-best' is a generalization of the classic 'DE/current-to-best' mutation strategy, in which not only the best-solution information is utilized, but also the information of other good solutions is utilized. In fact, any of the top $100p\%, p \in (0, 1]$, solutions can be randomly chosen in 'DE/current-to-p-best' to play the role that the single best solution plays in 'DE/current-to-best'. In order to diversify the population and avoid the premature convergence, the difference between the archived inferior solutions and the current population can be incorporated into the mutation operation. This

is an optional choice for JADE. Hence, the mutation utilized in JADE is called 'DE/current-to-p-best with/without external archive'. In this paper, we only consider the variant of JADE with external archive and still denote it as JADE. In this way, the mutation vector is generated in the following manner:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i(\mathbf{x}^p_{best,g} - \mathbf{x}_{i,g}) + F_i(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (6)$$

where $\mathbf{x}^p_{best,g}$ is a randomly chosen individual from the best $100p\%$ individuals in the current population, and F_i is the mutation scale factor associated with $\mathbf{x}_i$ and is regenerated at each generation by an adaptation strategy. $\mathbf{x}_{r1,g}$ is randomly selected from the union, $P \cup A$, where P represents current population and A represents the archive, while $\mathbf{x}_{i,g}$, $\mathbf{x}_{r1,g}$ and $\mathbf{x}^p_{best,g}$ are selected from P in the same way as in Eq. (2).

The archive operation is applied as follows: initially, A is set to be empty. After each generation, the parents failing in the selection process are added to A. If the archive size exceeds a predefined threshold, say NP, some solutions are randomly eliminated from A to keep the archive size at NP.

- Control parameter adaptation: In JADE, the crossover rate CR_i and mutation factor F_i associated with each individual vector $\mathbf{x}_i$ are generated using two parameters, μ_{CR} and μ_F, respectively. At each generation g, CR_i of each individual $\mathbf{x}_i$ is independently generated according to a normal distribution $randn_i(\mu_{CR}, 0.1)$ of mean μ_{CR} and standard deviation 0.1, and then truncated to [0,1]. Similarly, F_i of each individual $\mathbf{x}_i$ is independently generated according to a Cauchy distribution $randc_i(\mu_F, 0.1)$ with location parameter μ_F and scale parameter 0.1, and then truncated to the interval (0,1]. S_{CR} is denoted as the set of all *successful crossover rates* CR_i's at generation g. μ_{CR} is initialized to be 0.5 and then updated at the end of the generation as follows:

$$\mu_{CR} = (1-c)\mu_{CR} + c \cdot mean_A(S_{CR}) \quad (7)$$

where c is a constant between 0 and 1 and $mean_A(\cdot)$ is the arithmetic mean operation.

Similarly, S_F is denoted as the set of all *successful mutation factors* F_i's at generation g. μ_F is initialized to be 0.5 and then updated at the end of the generation as:

$$\mu_F = (1-c)\mu_F + c \cdot mean_L(S_F) \quad (8)$$

where $mean_L(\cdot)$ is the Lehmer mean:

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (9)$$

JADE has shown better or at least competitive performance in terms of both the convergence rate and the reliability, compared to other evolutionary algorithms. For detailed description of JADE, please refer to [3].

C. rJADE

We use the original JADE (with external archive) as our basic search procedure, and apply two strategies to it to form our approach, rJADE.

1) Modified control parameter adaptation: It can be referred that, by applying arithmetic mean in Eq. (7), the μ_{CR} of JADE has an implicit bias towards small values during self-adaptation process. To balance this bias, we make a slight modification on the original JADE by adding a weighting strategy on the adaptation of control parameter CR: whenever a successful CR value is recorded in the array S_{CR}, we will also record the corresponding fitness value improvement in array Δf_{rec}, with $\Delta f_{rec}(k) = f(k) - f_{new}(k)$. Note here $|S_{CR}| = |\Delta f_{rec}|$. Then, Eq. (7) is changed to:

$$\mu_{CR} = (1-c)\mu_{CR} + c \cdot \sum_{k=1}^{|S_{CR}|} \omega_k * S_{CR}(k) \quad (10)$$

$$\omega_k = \frac{\Delta f_{rec}(k)}{\sum_{k=1}^{|\Delta f_{rec}|} \Delta f_{rec}(k)} \quad (11)$$

2) Restart with knowledge transfer: In rJADE, whenever the current population converges at a local optimum, a restart is activated. Here, we regard a population as a converged one when the improvement of its best fitness value is equal to or less than a threshold δ_{fit} within certain generations $interval_{restart}$. Hence, as illustrated in Fig. 1, an independent run of rJADE will consist of several restarts. We call each restart a phase of the run. The information obtained from previous phase will be recorded into a tabu list T as knowledge. All the items in T will be transferred to the next phase to guide the subsequent search.

In our approach, the neighborhood area of the local optimum found in previous failure, specifically, its δ hypersphere neighborhood area, is treated as knowledge and will be recorded in T. Denote the best individual found in this situation as $\mathbf{x}_{best}$. Then the δ hypersphere neighborhood of this local optimum is defined by $[x_{best,j} - \delta, x_{best,j} + \delta]$, for $j = 1, 2, ..., D$. The newly generated population in the new phase will be forced away from previous local optima, i.e., no individual will be in any of the δ hypersphere neighborhood areas of previous local optima. In this way, the algorithm will be capable of searching away from previous local optima.

3) Perturbation: Sometimes some individuals of the current population (even the whole population) may fall into one of previous local optima again. The set of these individuals is denoted as V. If the number of individuals in V exceeds a threshold value, say δ_{vib}, we will apply a perturbation operation on the whole population and enable the population to search around this local optimum. The perturbation is operated as:

$$\mathbf{x}^{new}_i = \mathbf{x}^{old}_i + \lambda \cdot randn(0,1), \text{for } i = 1, 2, ..., NP \quad (12)$$

$\mathbf{x}^{old}_i$, $\mathbf{x}^{new}_i$ denote the old individual and new generated individual respectively. λ is a predefined number, and

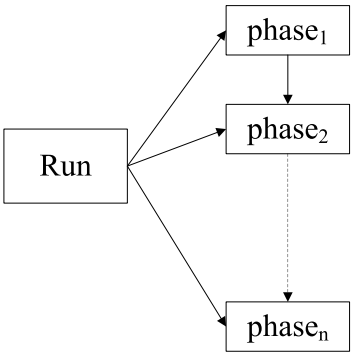

Fig. 1. Multi-phases scheme of rJADE

$randn(0,1)$ is a normal distribution with mean 0 and standard deviation 1.

After the perturbation, the archive used in JADE is reset to be empty as we do not want the individuals in the archive to influence the subsequent search. If the global optima locate close to the local optimum, by applying the perturbation, the population will be capable of searching around this local optimum.

It is worth noting that, we do not distinguish the global optima with local optima here. If the global optima are recorded into T, though the subsequent search is worthless and will not find the global optima again, it will not affect the reliability of the algorithm since we have already found the global optima.

The pseudo code of the proposed rJADE is given in Fig. 2.

```
1  Procedure: rJADE
2  Initialize: phase number i = 1, T = ∅, Generate the initial population, pop;
3              Evaluate all the individuals in pop;
4              fes = NP;
5  While fes < MAX_FES do
6      Evolve pop using JADE with modified CR adaptation strategy;
7      If T is not empty
8          Find all the individuals in Offspring that fall into T, denoted as V;
9          If |V| ≥ δ_vib
10             Apply a perturbation operation;
11             Evaluate the perturbed population;
12             fes = fes + NP;
13             Clear the external archive;
14         End if
15     End if
16     If no perturbation is applied
17         Apply the one-to-one selection;
18         fes = fes + NP;
19     End if
20     If pop converges to a local optimum
21         Record the δ neighborhood area of the local optimum into T;
22         Reinitialize the population with no individual being in the T;
23         Evaluate the new generated population;
24         fes = fes + NP;
25         Reset the control parameters;
26         Clear the external archive;
27         i = i + 1;
28     End if
29 End while
```

Fig. 2. Pseudo code of rJADE

III. Experimental Studies

A. Experimental Setup

As our main objective is to improve the reliability of JADE's performance and most of the unstable cases occur on multimodal problems, we choose one unimodal function and seven multimodal functions. Table 1 presents short descriptions of these functions with their specific features. More details can be found in the original references [20] and [21], respectively. For a fair comparison, we design the experiment as follows: we run the algorithm on a problem multiple times. For each run, the same maximum number of fitness evaluations, MAX_FES, is given. The algorithm terminates when it achieves a value that is equal to or less than a predefined tolerance value within the MAX_FES, or the MAX_FES is reached. The tolerance values chosen here are recommended by [3] and [22] and presented in the last column of Table 1. If the algorithm terminates before reaching MAX_FES, the corresponding run will be considered successful. The percentage of runs terminated successfully will be regarded as the success rate, and can be used to measure the reliability of the algorithm.

In our studies, rJADE is first compared with the original JADE where an external archive is maintained. Results of JADE were obtained using the code provided by the original author, and by applying the parameters suggested in the corresponding publication. For a fair comparison, we set the parameters of rJADE to be fixed, $\lambda = 5.0$, $\delta_{vib} = 0.1 * NP$, $\delta_{fit} = 0.01 * tol$, $interval_{restart} = 100$, $\delta = 0.001 * (UB - LB)$, where UB and LB denote the upper bound and lower bound of the initial range respectively, with the other parameters following the parameter settings in the original paper of JADE. To further demonstrate the effectiveness of the proposed rJADE, we also apply a "restart without knowledge transfer" strategy on the original JADE, and denote it as JADE/restart. Finally, rJADE is compared with a recent state-of-art evolutionary algorithm, IPOP-CMA-ES. All the functions were solved in 30-dimensions. To clearly demonstrate the reliability of the algorithms, 100 independent runs were conducted on each function.

B. Experimental Results and Discussions

The experimental results are examined from two aspects, the success rate (SR) and the shortest time required for a successful run. The former is a commonly used measure for comparing the reliability, while the latter is used to measure the convergence speed (in successful runs). Table 2 presents the success rate of each algorithm, together with the mean and standard deviation of function evaluations over successful runs, denoted as $MFES$ and $STDFES$ respectively. First, from the table we can observe that, for

TABLE I

TEST FUNCTIONS USED IN THIS STUDY, INCLUDING A SHORT DESCRIPTION OF THEIR CHARACTERISTICS. A DETAILED DESCRIPTION OF THE INDIVIDUAL FUNCTIONS, AND THEIR MATHEMATICAL EXPRESSION CAN BE FOUND IN [20] AND [21]. THE VALUES IN THE LAST COLUMN INDICATE THE TOLERANCE LIMITS USED FOR DEFINING A SUCCESSFUL RUN.

	UNIMODAL FUNCTIONS	Characteristics	Tolerance
f_5	Generalized Rosenbrock's Function	non-separable, scalable, narrow valley from local to global optimum	1.0e-8
f_8	Generalized Schwefel's Problem 2.26	separable, scalable, numerous local optima	1.0e-8
f_{11}	Generalized Griewank Function	non-separable, scalable, numerous local optima	1.0e-8
f_{13}	Generalized Penalized Function	separable, scalable, numerous local optima	1.0e-8
f_{cec2}	Shifted Schwefel's Problem 1.2	shifted, non-separable,scalable	1.0e-6
f_{cec6}	Shifted Rosenbrock's Function	shifted, non-separable, scalable, narrow valley from local to global optimum	1.0e-2
f_{cec7}	Shifted Rotated Griewank's Function without Bounds	rotated, shifted, non-separable, scalable	1.0e-2
f_{cec12}	Schwefel's Problem 2.13	shifted, non-separable, scalable	1.0e-2

TABLE II

SUCCESS RATE (SR) AND THE MEAN (STANDARD DEVIATION) OF FES NEEDED FOR A SUCCESSFUL RUN OF RJADE, JADE, JADE/RESTART, MJADE AND ROJADE.

Function		f_5	f_8	f_{11}	f_{13}	f_{cec2}	f_{cec6}	f_{cec7}	f_{cec12}
MAX_FES		500000	900000	300000	150000	300000	600000	300000	600000
rJADE	SR	1	1	1	1	1	1	1	0.23
	$MFES$	108484	87378	28186	26089	62120	104607	36326	248361
	$STDFES$	11015	14429	2879	840	3429	31120	23511	159056
JADE	SR	0.98	0.96	0.99	1	1	0.85	0.79	0.01
	$MFES$	114263	124355	34704	31508	64968	103539	32254	172100
	$STDFES$	4743	2652	4829	1066	3783	8191	4510	0
JADE/restart	SR	1	1	1	1	1	0.9	1	0.07
	$MFES$	117805	128650	34550	31503	64362	120686	42202	315843
	$STDFES$	22244	26569	4120	1101	4225	65470	23393	182817
mJADE	SR	0.96	0.97	0.99	1	1	0.95	0.79	0.07
	$MFES$	107789	85379	27631	25883	61544	94823	25396	119357
	$STDFES$	3916	2809	1226	748	3990	9152	2585	19801
roJADE	SR	1	1	1	1	1	0.91	1	0.07
	$MFES$	116828	131856	34882	31568	64754	126360	46338	265943
	$STDFES$	19973	33174	7280	1239	3825	65824	36602	160247
IPOP-CMA-ES	SR	1	0	1	1	1	1	0.99	0.22
	$MFES$	53033	—	3900	6711	13627	67293	7147	238504
	$STDFES$	10061	—	155	3527	394	30079	4487	20323

the problems where JADE can achieve a successful run with a high probability, i.e., f_5, f_8 and f_{11}, both of rJADE and JADE/restart achieved a success rate of 1.0. Since JADE achieves a high probability of finding the global optima, restarting the algorithm will obtain a great chance to achieve a success rate of 1.0. Hence, there will be no distinct difference between the results of rJADE and JADE/restart. For those problems where JADE obtained a positive but not big chance of achieving a successful run, rJADE increased the probability greatly. In detail, for functions f_{cec6}, f_{cec7} and f_{cec12}, rJADE enlarged the success rate from 0.85 to 1.0, 0.79 to 1.0, and 0.01 to 0.23, respectively. However, JADE/restart did not always significantly improve the reliability, especially on f_{cec6} and f_{cec12}. Since JADE has a chance to obtain a successful run, applying a restart strategy when the algorithm gets trapped in a local optimum might draw the current population out of the local optimum and thus enhance the success rate. Further, even when the restarted population falls around previous local optima in current phase, the perturbation operation will enable the algorithm to search around the local optima. This technique is especially useful when the global optima and local optima locate close. In general, rJADE outperformed both JADE and JADE/restart significantly on those problems where JADE showed unstable performance. Meanwhile, by examining the success rate of JADE and rJADE on f_{12} and f_{cec2}, we can find that for those problems on which JADE performed stably, rJADE did not deteriorate the performance.

Second, observing the $MFES$ and $STDFES$ of these two algorithms, no conclusive evidence can be found in Table 2. In detail, rJADE used less $MFES$ on f_5, f_8, f_{11}, f_{13} and f_{cec2}, while was outperformed by JADE on f_{cec6}, f_{cec7} and f_{cec12}. Similar results can be found between rJADE and JADE/restart. The loss of rJADE on some of these problems might be due to the restart strategy utilized when the algorithm got trapped in local optima. In this situation, JADE will fail in this run and the time for this run will be ignored when calculating the $MFES$. However, rJADE will activate a restarted phase and have a chance to succeed. If rJADE succeeds, the time for this run will be used for calculating the $MFES$. As restart is utilized, the total time used by rJADE in this run may exceed the time used by JADE

for a successful run. Hence, the final $MFES$ of rJADE may exceed JADE.

By analyzing both the reliability and convergence rate of the three algorithms, one may conclude that, by applying a "restart with knowledge transfer" strategy and the modified control parameter adaptation strategy, rJADE shows the most reliable or at least comparably reliable performance over all the test problems. Moreover, although the restart strategy is utilized, the convergence rate does not deteriorate significantly, which is also what we expected.

We also apply each one of the two proposed strategies on the original JADE: mJADE implements the modified control parameter adaptation strategy while roJADE implements the "restart with knowledge transfer" strategy. We find that none of the two algorithms achieved a satisfactory reliability of performance. As shown in Table 2, mJADE only achieved SR of 1 on functions f_{13} and f_{cec2}, where JADE also achieved SR of 1. roJADE improved the reliability of JADE on f_5, f_8, f_{11} and f_{cec7} as rJADE did. However, on f_{cec6} and f_{cec12}, roJADE showed less improvements than rJADE. rJADE outperforming both mJADE and roJADE indicates that, it is the combination of the two strategies that rJADE makes the significant improvements on the original JADE.

We finally compared rJADE with IPOP-CMA-ES on all the 8 test functions. From the last 3 rows of Table 2 we can observe that, rJADE showed superiorly better reliability than IPOP-CMA-ES on function f_8, where IPOP-CMA-ES only achieved $SR = 0$. rJADE also showed slightly better reliability than IPOP-CMA-ES on functions f_{cec7} and f_{cec12}. However, rJADE showed slower convergence speed on all the test functions. As the base algorithm we used to form our approach is JADE, rJADE showed similar convergence speed as JADE. Since JADE converged slower than IPOP-CMA-ES as shown in Table 2, it may not be surprising that IPOP-CMA-ES converged faster than our approach.

IV. Conclusions

Reliability is an important issue that must be paid enough attention to when designing an algorithm. In many real-world applications, simply restarting an algorithm when it fails may lead to an inefficient or even ineffective search. In this case, recording the information obtained from previous failures and using it as the knowledge to guide the subsequent search will be a good choice. In this paper, we proposed a new algorithm named rJADE, which introduces two strategies to the original JADE to enhance its reliability. In rJADE, we first modify the control parameter adaptation strategy of JADE by adding a weighting strategy. Then, a "restart with knowledge transfer" strategy is applied: whenever the current population converges at a local optimum, a restart is activated. The areas around the local optima found in previous failures are recorded as knowledge, and will be transferred to next phase to guide the subsequent search. A perturbation operation is also applied in case of the restarted population falling into previous local optima again. Hence, our approach would be capable of searching areas both distant from and around previous local optima. Empirical results showed significant improvements on reliability of the original JADE. Further, by comparing the performance of rJADE with a restart JADE without knowledge transfer, we can find the superiority of modifying the control parameter adaptation strategy and using the knowledge to guide the search process. We also demonstrate that, only by combining the two proposed strategies together can rJADE achieve the most significant improvements on the original JADE. Finally, we compared rJADE with a state-of-art evolutionary algorithm, IPOP-CMA-ES, and obtained better or at least comparable reliability than IPOP-CMA-ES on all the test functions.

A few issues regarding rJADE remain to be further investigated in the future. One is to extend the idea of the "restart with knowledge transfer" scheme to other EAs to enhance the reliability. Another one would be to make a modification on the neighborhood definition which is still simply defined in our approach. The perturbation used in our approach is rather simple and straightforward, and may not perform well. Alternative approaches for perturbation deserve further study. Finally, effects of the parameters used in rJADE should be analyzed as well.

Acknowledgment

This work is partially supported by a National Natural Science Foundation of China grant (No. 60533020), the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033), and the Fund for International Joint Research Program of Anhui Science & Technology Department (No. 08080703016).

References

[1] R. Storn and K. Price, "Differential Evolution - a simple and efficient heuristic strategy for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[2] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, ISBN: 3–540–20950–6, 2005.

[3] J. Zhang and A. Sanderson, "JADE: adaptive differential evolution with optional external archive," Accepted by *IEEE Transactions on Evolutionary Computation*, May. 2008.

[4] T. Bäck, "Evolutionary algorithms in theory and practice," Oxford University Press, New York, 1996.

[5] T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation: The New Experimentalism*. Berlin Heidelberg: Springer-Verlag, 2006.

[6] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[7] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 561–578, Oct. 2007.

[8] N. Noman and H. Iba, "Accelerating Differential Evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, Feb. 2008.

[9] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," *in Proceedings of the 1998 IEEE Congress on Evolutionary Computation (CEC'98)*, pp. 69–73, Anchorage, AK, USA, 1998.

[10] Z. Yang, K. Tang, and X. Yao, "Self-adaptive Differential Evolution with Neighborhood Search," *in Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC'08)*, pp. 1110–1116, Hongkong, 2008.

[11] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[12] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[13] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *in Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'05)*, vol. 2, pp. 1785–1791, Edinburgh, UK, 2005.

[14] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[15] Y. Wang and B. Li, "A restart univariate estimation of distribution algorithms: sampling under mixed Gaussian and Lévy probability distribution," *in Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC'08)*, pp. 3917–3924, Hongkong, 2008.

[16] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," *in Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'05)*, vol. 2, pp. 1769–1776, Edinburgh, UK, 2005.

[17] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, vol. 8, no. 1, pp. 156–166, 1977.

[18] T. A. A. Victoire and A. E. Jeyakumar, "Unit commitment by a tabu-search-based hybrid-optimisation technique," *IEE Proceedings of Generation, Transmission and Distribution*, vol. 152, no. 4, pp. 563–574, 2005.

[19] Y. Wang, Z. Zhao, and R. Ren, "Hybrid particle swarm optimizer with Tabu strategy for global numerical optimization," *in Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC'07)*, pp. 2310–2316, Singapore, 2007.

[20] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[21] P. N. Suganthan et. al, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," http://www.ntu.edu.sg/home/EPNSugan, 2005.

[22] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," Accepted by *IEEE Transactions on Evolutionary Computation*, Feb. 2008.