WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

The Continuous Differential Ant-Stigmergy Algorithm Applied to Dynamic Optimization Problems

Peter Korošec, *Member, IEEE,* Jurij Šilc, *Member, IEEE,*

***Abstract*— Many real-world problems are dynamic, requiring an optimization algorithm which is able to continuously track a changing optimum over time. In this paper, we present an ant-colony based algorithm for solving optimization problems with continuous variables, labeled Continuous Differential Ant-Stigmergy Algorithm (CDASA). The CDASA is applied to dynamic optimization problems without any modification to the algorithm. The performance of the CDASA is evaluated on the set of benchmark problems provided for the IEEE Competition on Evolutionary Computation for Dynamic Optimization Problems (ECDOP-Competition-2012).**

I. Introduction

A dynamic optimization problem (DOP), which this work deals with, is defined as

$$F = f(x, \phi, t),$$

where F is the optimization problem, f is the fitness function, x is a feasible solution in the solution set $\mathbf{X}$, t is time and ϕ is the system control parameter, which determines the solution distribution in the fitness landscape.

Recently, there have been many attempts to solve these problems using evolutionary algorithms [1], differential evolution [5], [6], memetic algorithms [2], multi-agent algorithms [3], particle swarm optimization [4], and ant-colony optimization [7], [6]. Cruz et. al. [8] provide a survey of the research done on optimization in dynamic environments over the past decade.

In this paper we describe the Continuous Differential Ant-Stigmergy Algorithm (CDASA) for dynamic optimization based on idea of continuous space exploration with probabilistic sampling. Moreover, the CDASA is modification of the Differential Ant-Stigmergy Algorithm (DASA) proposed in [9], [10], which transforms a real-parameter optimization problem into a graph-search problem, where vertices represent predefined offsets. Unlike the DASA, the CDASA uses arbitrary real offsets to navigate through the search space.

The remainder of this paper is organized as follows: Section II introduces the optimization algorithm called the Differential Ant-Stigmergy Algorithm. Section III presents the experimental evaluation on the set of benchmark problems provided for IEEE Competition on Evolutionary Computation for Dynamic Optimization Problems (ECDOP-Competition-2012). Finally, Section IV concludes the paper.

Peter Korošec, peter.korosec@ijs.si (corresponding author), and Jurij Šilc, jurij.silc@ijs.si, are with the Jožef Stefan Institute, Ljubljana, Slovenia.

II. The Algorithm

Let us assume n is a number of parameters x_i, $i = 1, 2, \ldots, n$, and x'_i is the current value of the i-th variable. During the search for the optimal variable value, the new value, x_i, is assigned to the i-th variable as follows:

$$x_i = x'_i + \Delta_i. \tag{1}$$

Here, Δ_i is the so-called *variable difference* and is chosen using the inverse cumulative distribution function of the Cauchy distribution:

$$\Delta_i = z_{0_i} + s \tan(Rnd_i),$$

where z_{0_i} is the location offset for the i-th variable, s is the scale factor, and Rnd_i is a uniformly distributed random number from continuous interval, so for all $\Delta_i, i = 1, 2, \ldots, n$, $l_i \leq x_i + \Delta_i \leq u_i$ is true.

Each ant walks over all Cauchy distributions and on its way forms the vector of variable differences

$$\vec{\Delta} = [\Delta_1 \ \Delta_2 \ \ldots \ \Delta_i \ \ldots \ \Delta_n].$$

The optimization consists of an iterative improvement of the temporary best solution, $\vec{x}^{\,\text{tb}}$, by constructing an appropriate vector of variable differences, $\vec{\Delta}$. New solutions are produced by applying $\vec{\Delta}$ to $\vec{x}^{\,\text{tb}}$ (Eq. 1).

First a solution $\vec{x}^{\,\text{tb}}$ is randomly chosen (by uniform sampling) in `RndSolution`. This solution is then evaluated and set as global best, y^{gb}. Then an initial amount of pheromone is deposited on variable difference intervals according to the Cauchy probability density function

$$C(z; z_{0_i}, s) = \frac{1}{\pi}\left(\frac{s}{(z - z_{0_i})^2 + s^2}\right),$$

where z_{0_i} is the location offset for the i-th variable and

$$s = s_{\text{global}} - s_{\text{local}}$$

is the scale factor. For an initial pheromone distribution implemented in `PheromoneInitialization` the Cauchy distribution with $s_{\text{global}} = 1$, $s_{\text{local}} = 0$, and $z_{0_i} = 0$, for $i = 1, 2, \ldots, n$ is used.

There are m ants in a colony, all of which begin simultaneously from the first variable. Ants use an inverse cumulative distribution function to determine a variable difference.

The ants repeat this action until they reach the last variable. For each ant i, variable differences vector $\vec{\Delta}_i$ is constructed in `FindDifferences` and from which a new solution $\vec{x_i}$ is calculated.

After all ants have created solutions, they are being evaluated with a calculation of $y = f(\vec{x_i})$. The information about the best among them is stored as currently best information ($\vec{x}^{\text{ cb}}$, $\vec{\Delta}^{\text{ cb}}$, and $y^{\text{ cb}}$).

The current best solution, $\vec{x}^{\text{ cb}}$ is compared to the temporary best solution $\vec{x}^{\text{ tb}}$. If $y^{\text{ cb}}$ is better than $y^{\text{ tb}}$, then temporally best information is replaced with currently best information. In this case s_{global} is increased according to the global scale increase factor, s_+:

$$s_{\text{global}} \leftarrow (1+s_+)s_{\text{global}},$$

s_{local} is set to

$$s_{\text{local}} = \frac{1}{2}s_{\text{global}}$$

in `UpdateScales` and pheromone amount is redistributed in `PheromoneRedistribution` according to the associated variable differences vector $\vec{\Delta}^{\text{ cb}}$, where $z_{0_i} = \Delta_i^{\text{cb}}$ for $i = 1, 2, \ldots, n$. Furthermore, if new $y^{\text{ tb}}$ is better then global best $y^{\text{ gb}}$, then globally best information is replaced with temporally best information. So, global best solution is stored. If no better solution is found s_{global} is decreased according to the global scale decrease factor, s_-:

$$s_{\text{global}} \leftarrow (1-s_-)s_{\text{global}}$$

in `UpdateScale` and pheromone is being evaporated in `PheromoneEvaporation`.

Pheromone evaporation is defined by some predetermined percentage ρ. The probability density function $C(z; z_{0_i}, s)$ is changed in the following way:

$$z_{0_i} \leftarrow (1-\rho)z_{0_i}$$

and

$$s_{\text{local}} \leftarrow (1-\rho)s_{\text{local}}.$$

Here we must emphasize that $\rho > s_-$, because otherwise we might get negative scale factor.

Finally, if scale factor s is smaller than restart threshold α, a new random solution is set as new temporary best and pheromone is initialized according to initial pheromone distribution.

The pseudocode of the CDASA is as follows:

```
$\vec{x}^{\text{ tb}}$ = RndSolution()
$y^{\text{ tb}} = \infty$
$y^{\text{ gb}} = f(\vec{x}^{\text{ tb}})$
PheromoneInitialization()
while not ending condition met do
  for all $m$ ants do
    $\vec{\Delta}_i$ = FindDifferences()
    $\vec{x}_i = \vec{x}^{\text{ tb}} + \vec{\Delta}_i$
  end for
  $y^{\text{ cb}} = \infty$
  for all $m$ ants do
    $y = f(\vec{x}_i)$
    if $y < y^{\text{ cb}}$ then
      $\vec{x}^{\text{ cb}} = \vec{x}_i$
      $y^{\text{ cb}} = y$
      $\vec{\Delta}^{\text{ cb}} = \vec{\Delta}_i$
    end if
  end for
  if $y^{\text{ cb}} < y^{\text{ tb}}$ then
    $\vec{x}^{\text{ tb}} = \vec{x}^{\text{ cb}}$
    $y^{\text{ tb}} = y^{\text{ cb}}$
    $s$ = UpdateScales($s_{\text{global}}, s_{\text{local}}$)
    PheromoneRedistribution($\vec{\Delta}^{\text{ cb}}, s$)
    if $y^{\text{ tb}} < y^{\text{ gb}}$ then
      $\vec{x}^{\text{ gb}} = \vec{x}^{\text{ tb}}$
      $y^{\text{ gb}} = y^{\text{ tb}}$
    end if
  else
    UpdateScale($s_{\text{global}}$)
    PheromoneEvaporation($\rho$)
  end if
  if $s < \alpha$ then
    $\vec{x}^{\text{ tb}}$ = RndSolution()
    $y^{\text{ tb}} = f(\vec{x}^{\text{ tb}})$
    PheromoneInitialization()
  end if
end while
```

III. Performance Evaluation

A. The Benchmark Suite

The CDASA algorithm was tested on six benchmark problems provided for the IEEE Competition on Evolutionary Computation for Dynamic Optimization Problems (ECDOP-Competition-2012) [11]:

- F_1: Rotation peak function (multi-modal, scalable, rotated, the number of local optima are artificially controlled),
- F_2: Composition of Sphere's function (multi-modal, scalable, rotated, 10 local optima),
- F_3: Composition of Rastrigin's function (multi-modal, scalable, rotated, a huge number of local optima),
- F_4: Composition of Griewank's function (multi-modal, scalable, rotated, a huge number of local optima),
- F_5: Composition of Ackley's function (multi-modal, scalable, rotated, a huge number of local optima),
- F_6: Hybrid composition function (multi-modal, scalable, rotated, a huge number of local optima, different functions properties are mixed together, sphere functions give two flat areas for the function).

B. Framework of Dynamic Changes

When dealing with the dynamic optimization problem the dynamism results from a deviation of solution distribution from the current environment by tuning the system control parameters. It can be described as follows:

$$\phi(t+1) = \phi(t) \oplus \Delta\phi,$$

where $\Delta\phi$ is a deviation from the current system control parameters. Then, we can get the new environment at the next moment $t+1$ as follows:

$$f(x, \phi(t+1)) = f(x, \phi(t) \oplus \Delta\phi).$$

There are eight change types of the system control parameters:

- T_1: small step change
$$\Delta\phi = \alpha \cdot \|\phi\| \cdot r \cdot \phi_{\text{severity}},$$
- T_2: large step change
$$\Delta\phi = \|\phi\| \cdot (\alpha \cdot \text{sign}(r) + (\alpha_{\max} - \alpha) \cdot r) \cdot \phi_{\text{severity}},$$
- T_3: random change
$$\Delta\phi = N(0,1) \cdot \phi_{\text{severity}},$$
- T_4: chaotic change
$$\phi(t+1) = A \cdot (\phi(t) - \phi_{\min}) \cdot \frac{1 - (\phi(t) - \phi_{\min})}{\|\phi\|},$$
- T_5: recurrent change
$$\phi(t+1) = \phi_{\min} + \frac{\|\phi\|}{2} \cdot (\sin(\frac{2\pi}{P}t + \varphi) + 1),$$
- T_6: recurrent change with noise
$$\phi(t+1) = \phi_{\min} + \frac{\|\phi\|}{2} \cdot (\sin(\frac{2\pi}{p}t + \varphi) + 1) \cdot N_{\text{severity}},$$
- T_7: dimensional change
$$D(t+1) = D(t) + \text{sign} \cdot \Delta D,$$
and
- T_8: number of peaks change
$$P(t+1) = P(t) + \text{sign} \cdot \Delta P.$$

Here, $\|\phi\|$ is the change range of ϕ, ϕ_{severity} is a constant number that indicates change severity of ϕ, $\phi_{\min}$ is the minimum value of ϕ, $N_{\text{severity}} \in (0,1)$ is noisy severity in recurrent with noisy change and is set to 0.8. $\alpha \in (0,1)$ and $\alpha_{\max} \in (0,1)$ are constant values, which are set to 0.04 and 0.1. A logistics function is used in the chaotic change type, where $A \in (1.0, 4.0)$ is a positive chaotic constant, which is set to 3.67, if ϕ is a vector, the initial values of the items in ϕ should be different within $\|\phi\|$ in chaotic change. p is the period of recurrent change and recurrent change with noise and is set to 12, φ is the initial phase, r is a random number in $(-1,1)$, $\text{sign}(x)$ returns 1 when x is greater than 0, returns -1 when x is less than 0, otherwise, returns 0. $N(0,1)$ denotes a normally distributed one dimensional random number with mean zero and standard deviation one. ΔD is a predefined constant, which the default value of is 1. If $D(t) = \text{Max\_D}$, $sign = -1$; if $D(t) = \text{Min\_D}$, $\text{sign} = 1$. Max_D and Min_D are the maximum and minimum number of dimensions, which are in our case set to 15 and 2, respectively. ΔP is also a predefined constant, which the default value of is 2. If $P(t) = \text{Max\_P}$, $sign = -1$; if $P(t) = \text{Min\_P}$, $\text{sign} = 1$. Max_P and Min_P are the maximum and minimum number of peaks, which are in our case set to 50 and 10, respectively.

C. Parameter Settings

The CDASA has five parameters: the number of ants, m, the pheromone evaporation factor, ρ, the global scale increase factor, s_+, the global scale decrease factor, s_-, and the restart threshold, α. The single setting for parameters' was used for all problems. We set $m = 1$, $\rho = 0.91$, $s_+ = 0.03$, $s_- = 0.03$, and $\alpha = 13^{-4}$. The algorithm parameter settings were set according to the algorithmic approach to parameter tuning proposed by Smit and Eiben [12].

D. The Experimental Environment

The computer platform used to perform the experiments was based on AMD Opteron 2.6-GHz processor, 2 GB of RAM, and the Microsoft Windows operating system. The CDASA was implemented in C++ using the EAlib package, which is available from `http://cs.cug.edu.cn/teacherweb/lichanghe/pages/EAlib.html`.

E. Results

For each change type of each function, we present mean value (Mean) and standard deviation (StD) for $x^{\text{b}}(t)$ over 20 runs:
$$\text{Mean} = \sum_{i=1}^{\text{runs}} \sum_{j=1}^{\text{num\_change}} \frac{E_{i,j}^{\text{last}}(t)}{\text{runs} \cdot \text{num\_change}},$$
$$\text{StD} = \sqrt{\frac{1}{\text{runs} \cdot \text{num\_change}} \sum_{i=1}^{\text{runs}} \sum_{j=1}^{\text{num\_change}} (E_{i,j}^{\text{last}}(t) - \text{Mean})^2}.$$
Here, $E^{\text{last}}(t) = |f(x^{\text{b}}(t)) - f(x^*(t))|$ after reaching Max_FES/change for each change (in our case we have Max_FES/change = 50,000).

In Tables I–II we present the Mean values and StD for the CDASA algorithm on all test problems.

F. Overall Performance Marking Measurement

To evaluate the CDASA performance in terms of both convergence speed and solution quality, the performance on test case k is calculated by:
$$\text{performance}_k = \sum_{i=1}^{\text{runs}} \sum_{j=1}^{\text{num\_change}} \frac{r_{ij}}{\text{num\_change} * \text{runs}},$$
where $r_{ij} = \frac{r_{ij}^{\text{last}}}{1 + \sum_{s=1}^{S} \frac{1 - r_{ij}^s}{S}}$, r_{ij}^{last} is the relative value of the best one to the global optimum after reaching Max_FES/change for each change. r_{ij}^s is the relative value of the best one to the global optimum at the s-th sampling during one change and $S = \frac{\text{Max\_FES/change}}{s_f}$, where s_f is sampling frequency (in our case we have $s_f = 100$). $r_{ij}^s = \frac{f(x_{ij}) + \varepsilon}{f(x_{ij}^*) + \varepsilon}$ for the maximization problem F_1 and $r_{ij}^s = \frac{f(x_{ij}^*) + \varepsilon}{f(x_{ij}) + \varepsilon}$ for the minimizations problems F_2 – F_6, here ε is used to ensure that $f(x_{ij}^*) + \varepsilon > 0$.

The overall algorithm performance is evaluated by:
$$\text{performance} = \sum_{k=1}^{\text{Num\_test\_cases}} \text{performance}_k.$$

TABLE I

MEAN VALUES AND STD FOR THE PROBLEM F_1 WITH CHANGING RATIOS $\text{change\_ratio} = 0.3, 0.7,$ AND 1.0, RESPECTIVELY

Chang. ratio	T_1	T_2	T_3	T_4
	Mean ± StD	Mean ± StD	Mean ± StD	Mean ± StD
0.3	7.70E−1 ± 4.72E−1	1.69E+0 ± 9.25E−1	2.43E−1 ± 1.96E−1	3.46E−1 ± 5.12E−1
0.7	8.39E+0 ± 7.17E+0	4.45E+0 ± 3.16E+0	3.88E−1 ± 3.80E−1	3.20E−1 ± 3.33E−1
1.0	3.43E+0 ± 3.80E+0	3.08E+0 ± 2.47E+0	2.36E+0 ± 6.90E−1	9.76E−3 ± 2.30E−3
Chang. ratio	T_5	T_6	T_7	T_8
	Mean ± StD	Mean ± StD	Mean ± StD	Mean ± StD
0.3	2.81E−4 ± 1.36E−4	1.29E+0 ± 1.95E+0	—	—
0.7	2.08E−4 ± 9.03E−5	1.53E−1 ± 1.34E−1	—	—
1.0	1.68E−4 ± 5.74E−5	1.88E−2 ± 2.44E−3	2.63E+0 ± 4.84E+0	2.97E+0 ± 5.64E+0

TABLE II

MEAN VALUES AND STD FOR THE PROBLEMS F_2 – F_6

Problem	T_1	T_2	T_3	T_4
	Mean ± StD	Mean ± StD	Mean ± StD	Mean ± StD
F_2	5.44E−1 ± 5.10E−1	3.37E−1 ± 5.31E−1	1.13E+0 ± 1.66E+0	5.44E−2 ± 2.34E−2
F_3	7.53E+0 ± 1.18E+1	4.53E+1 ± 1.03E+2	4.05E+1 ± 9.10E+1	2.81E+1 ± 6.35E+1
F_4	1.90E+0 ± 3.49E+0	1.60E+0 ± 2.39E+0	3.46E+0 ± 4.37E+0	8.49E−1 ± 1.53E+0
F_5	1.90E−1 ± 9.47E−2	1.79E−1 ± 4.73E−2	1.86E−1 ± 1.10E−1	1.65E−1 ± 5.91E−2
F_6	1.39E+0 ± 1.94E+0	3.09E+0 ± 1.97E+0	2.84E+0 ± 3.64E+0	1.88E+0 ± 1.36E+0
Problem	T_5	T_6	T_7	T_8
	Mean ± StD	Mean ± StD	Mean ± StD	Mean ± StD
F_2	1.86E+0 ± 1.59E+0	1.47E−1 ± 2.31E−1	1.81E+0 ± 4.32E+0	8.18E+0 ± 1.10E+1
F_3	1.47E+1 ± 4.30E+1	3.15E+1 ± 6.59E+1	5.69E+2 ± 4.22E+2	1.53E+1 ± 1.92E+1
F_4	7.25E+0 ± 1.03E+1	1.10E+0 ± 1.73E+0	3.68E+0 ± 9.73E+0	1.70E+1 ± 3.85E+1
F_5	2.80E−1 ± 9.08E−2	1.59E−1 ± 4.79E−2	4.11E−1 ± 6.83E−1	8.65E−1 ± 9.18E−1
F_6	1.05E+0 ± 8.00E−1	1.96E+0 ± 1.13E+0	4.28E+0 ± 7.41E+0	7.95E+0 ± 8.28E+0

TABLE III

OVERALL PERFORMANCE

Problem	Chang. ratio	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Sum
F_1	0.3	0.9896	0.9809	0.9950	0.9819	0.9985	0.9636	—	—	5.9095
	0.7	0.8819	0.9450	0.9894	0.9913	0.9982	0.9910	—	—	5.7968
	1.0	0.9411	0.9416	0.9632	0.9924	0.9989	0.9895	0.9402	0.9422	7.7093
F_2	1.0	0.9495	0.9545	0.9366	0.9652	0.8710	0.9568	0.9011	0.7716	7.3062
F_3	1.0	0.7126	0.5948	0.6243	0.6015	0.7129	0.5991	0.4811	0.6129	4.9392
F_4	1.0	0.8640	0.8695	0.8467	0.8685	0.7712	0.8595	0.8537	0.7043	6.6375
F_5	1.0	0.9612	0.9605	0.9624	0.9521	0.9501	0.9531	0.9293	0.9079	7.5766
F_6	1.0	0.9074	0.8763	0.8771	0.8533	0.9130	0.8510	0.8351	0.7460	6.8593
The overall performance										52.7344

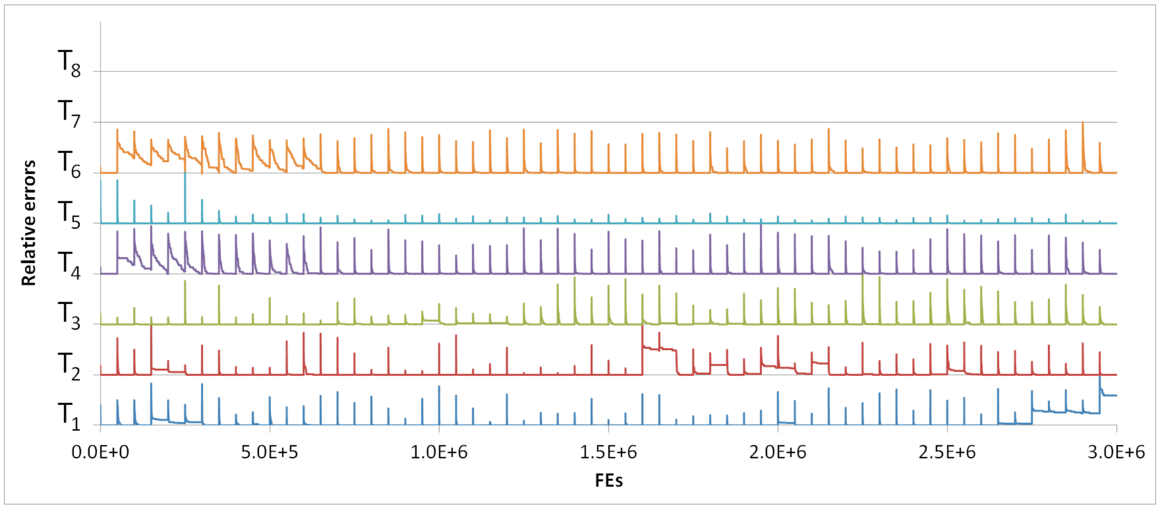

Fig. 1. Convergence graph for F_1 with changing ratio 0.3 for different change types.

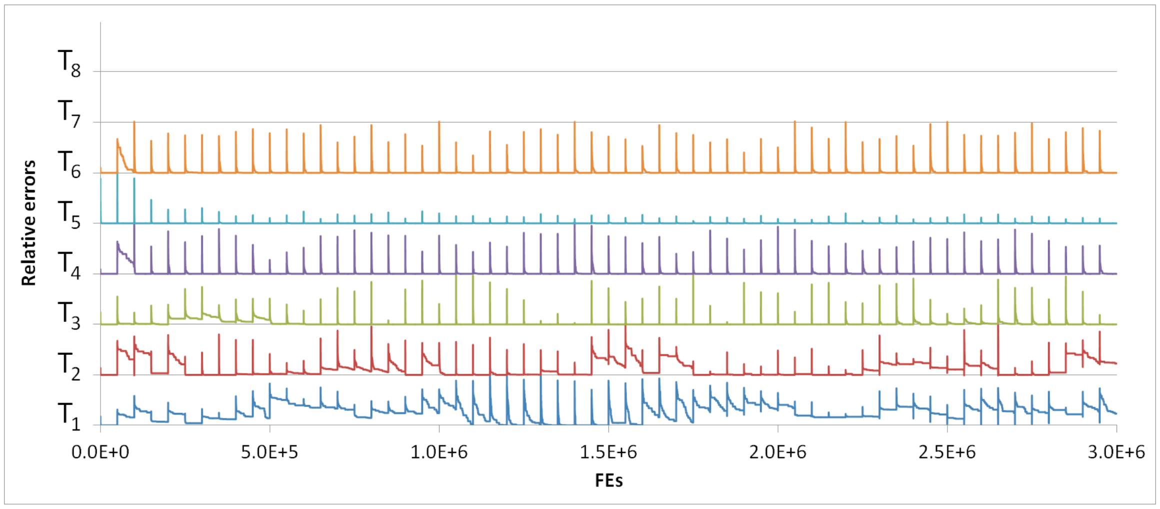

Fig. 2. Convergence graph for F_1 with changing ratio 0.7 for different change types.

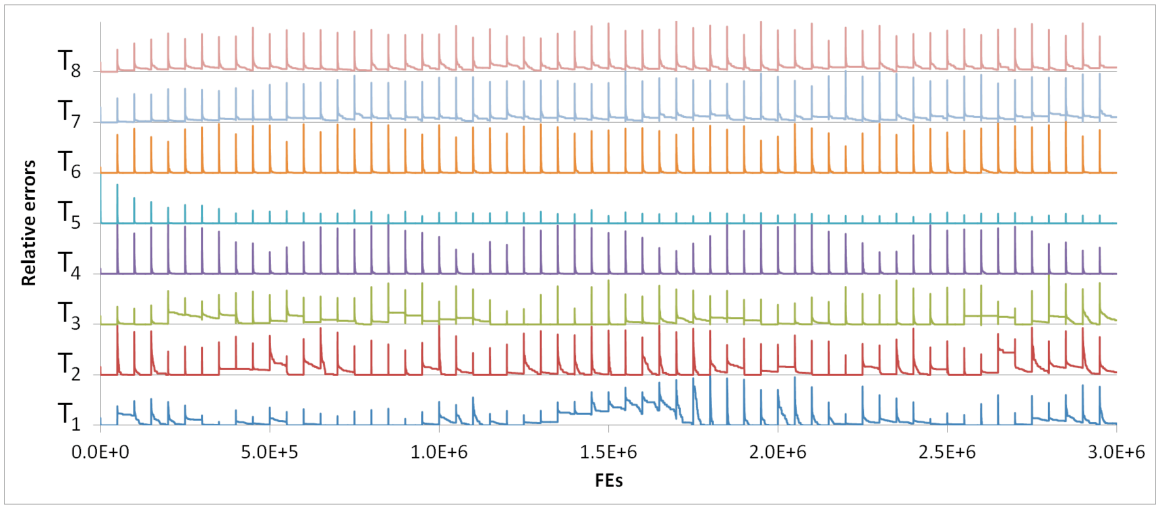

Fig. 3. Convergence graph for F_1 with changing ratio 1.0 for different change types.

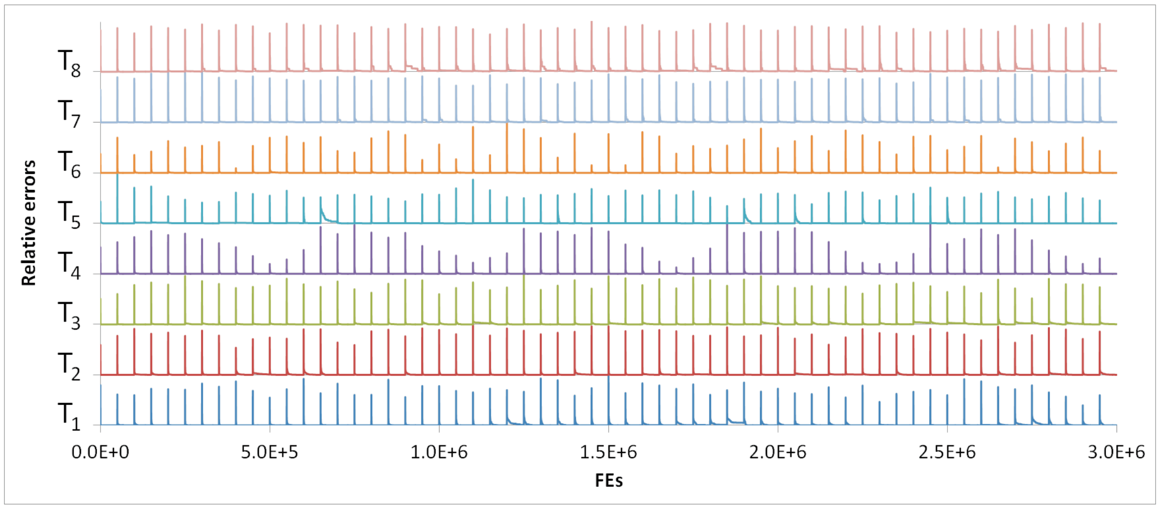

Fig. 4. Convergence graph for F_2 for different change types.

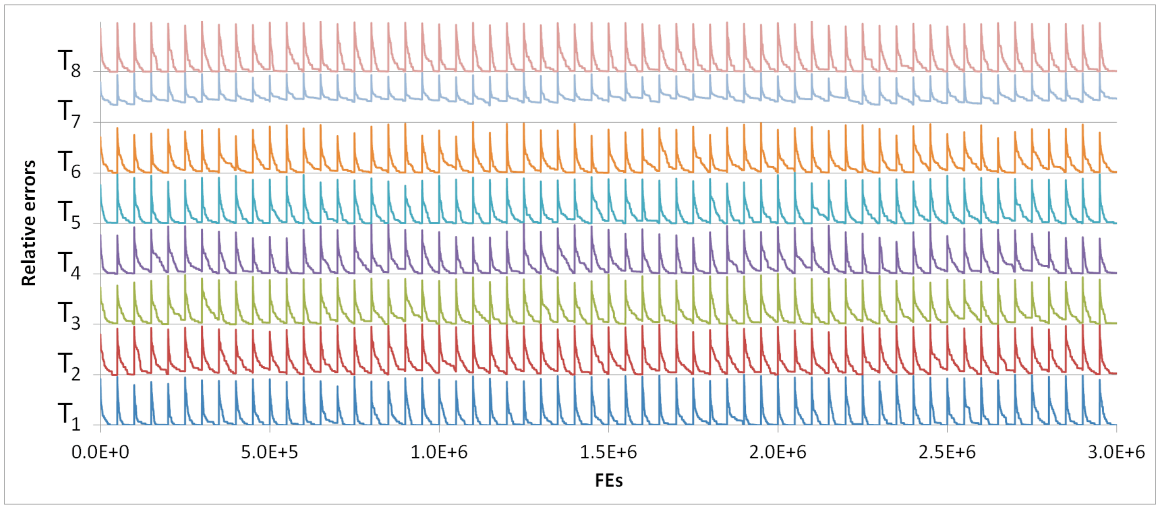

Fig. 5. Convergence graph for F_3 for different change types.

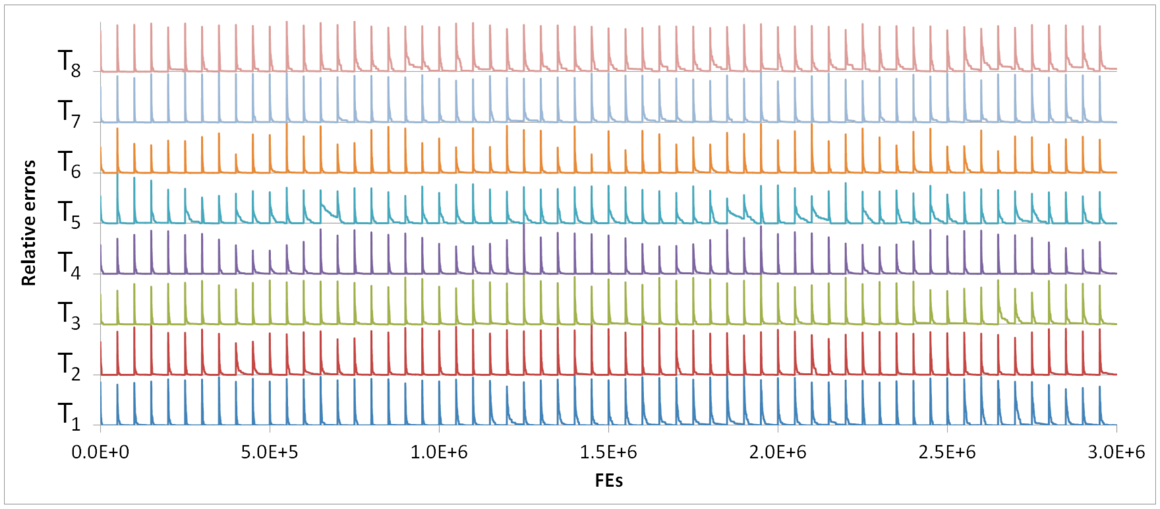

Fig. 6. Convergence graph for F_4 for different change types.

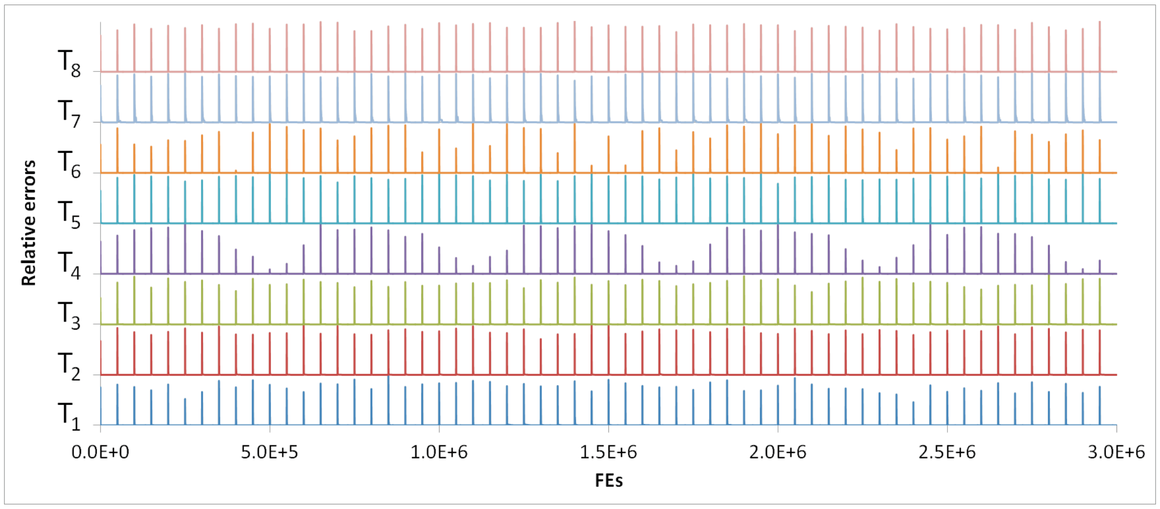

Fig. 7. Convergence graph for F_5 for different change types.

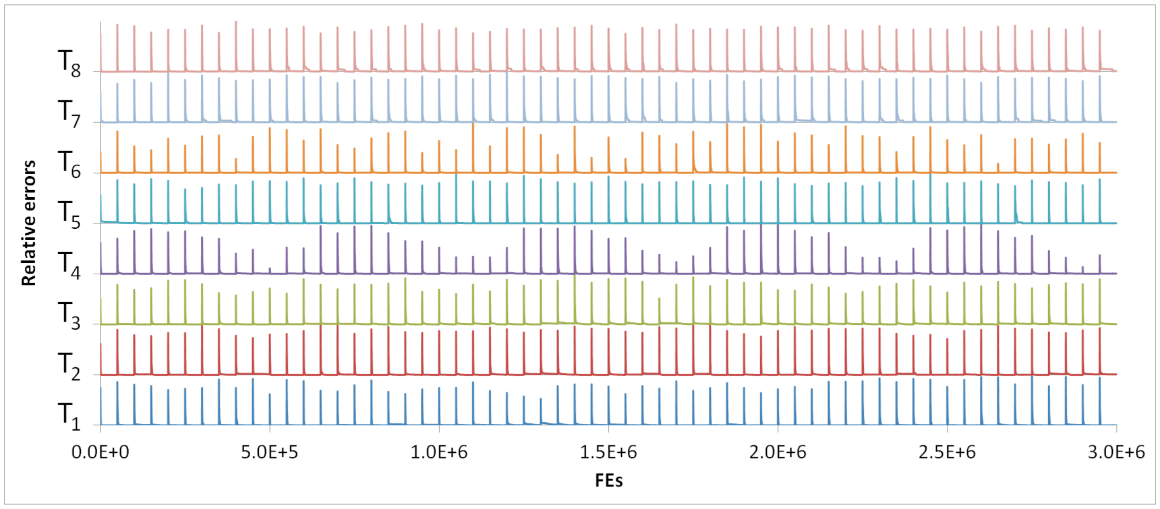

Fig. 8. Convergence graph for F_6 for different change types.

There are totally $\text{Num\_test\_cases} = 60$ specific test cases.

Table III presents the performance of the CDASA algorithm.

Convergence graphs for all functions obtained by the CDASA are depicted in Figs. 1–8. Normalized relative error values according to worst error obtained in all function evaluations are depicted in figures. This way all values are bijectively transformed to the interval $[0, 1]$. In general, the algorithm performed well on all test cases. The only real exception is function F_3 with dimensional change (T_7) and to some extend function F_1 with small and large step change.

IV. Conclusion

This paper presented an ant-colony based algorithm developed for numerical optimization problems. The algorithm was applied to dynamic optimization problems with continuous variables proposed for the IEEE Competition on Evolutionary Computation for Dynamic Optimization Problems (ECDOP-Competition-2012). The results showed that the proposed algorithm can find reasonable solutions for all of the problems.

One obvious advantage is that was no need any changes to the original algorithm. So, it can be used as such for both cases of numerical optimization, static and dynamic. From our previous testing [13] we have discovered that for static problems a higher number of ants is more suitable, while for dynamic problem a much lower number is needed, so the algorithm responds quicker to the changes. The same was true for the original algorithm DASA, where we have similarly used smaller number of ants to solve dynamic optimization problems [14]. We believe that one of the crucial reasons that there was no need to change the algorithm itself was in restart procedure, which inherently added the capacity to recover quickly every time there was a change in problem environment. Furthermore, the algorithm is unsusceptible to different types of changes and can be used with very limited knowledge about problem, only maximal dimension and input problem parameters.

References

[1] Y. Jin and J. Branke, “Evolutionary optimisation in uncertain environments a survey,” *IEEE Transaction on Evolutionary Computation*, vol. 9, pp. 303–317, 2005.

[2] H. Wang, D. Wang, S. and Yang, “A memetic algorithm with adaptive hill climbing strategy for dynamic optimisation problems,” *Soft Computing*, vol. 13, pp. 763–780, 2009.

[3] J. Lepagnot, A. Nakib, H. Oulhadj, and P. Siarry, “A new multiagent algorithm for dynamic continuous optimization,” *International Journal of Applied Metaheuristic Computing*, vol. 1, pp. 16–38, 2010.

[4] S. Yang and C. Li, “A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments,” *IEEE Transaction on Evolutionary Computation*, vol. 14, pp. 959–974, 2010.

[5] A. E. Kanlikilicer, A. Keles, and A. S. Uyar, “Experimental analysis of binary differential evolution in dynamic environments,” *Proc. Genetic and Evolutionary Computation Conference*, London, UK, July 2007, pp. 2509–2514.

[6] J. Brest, P. Korošec, J. Šilc, A. Zamuda, B. Bošković, and M. Sepesy Maučsec, “Differential evolution and differential ant-stigmergy on dynamic optimisation problems,” *International Journal of Systems Science*, doi:10.1080/00207721.2011.617899, 2012.

[7] W. Tfaili, J. Dréo, and P. Siarry, “Fitting of an ant colony approach to dynamic optimization through a new set of test functions,” *International Journal of Computational Intelligence Research*, vol. 3, pp. 203–216, 2007.

[8] C. Cruz, J. Gonzlez, and D. Pelta, “Optimization in dynamic environments: a survey on problems, methods and measures,” *Soft Computing*, vol. 15, pp. 1427–1448, 2011.

[9] P. Korošec and J. Šilc, “High-dimensional real-parameter optimization using the differential ant-stigmergy algorithm,” *International Journal of Intelligent Computing and Cybernetics*, vol. 2, pp. 34–51, 2009.

[10] P. Korošec, J. Šilc, and B. Filipič, “The differential ant-stigmergy algorithm,” *Information Sciences*, vol. 192, pp. 82–97, 2012.

[11] C. Li, S. Yang, D. A. Pelta, *Benchmark Generator for the IEEE WCCI-2012 Competition on Evolutionary Computation for Dynamic Optimization Problems*, October 28, 2011. `http://people.brunel.ac.uk/~csstssy/ECiDUE/TR-ECDOP-Competition12.pdf`

[12] S. Smit and A. Eiben, “Comparing parameter tuning methods for evolutionary algorithms,” *Proc. 2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, 2009, pp. 399–406.

[13] P. Korošec and J. Šilc, “The continuous differential ant-stigmergy algorithm applied to real-world optimization problems,” *Proc. 2011 IEEE Congress on Evolutionary Computation*, New Orleans, Lousiana, USA, 2011, pp. 1327–1334 .

[14] P. Korošec and J. Šilc, “The differential ant-stigmergy algorithm applied to dynamic optimization problems,” *Proc. 2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, 2009, pp. 407–414.