The Differential Ant-Stigmergy Algorithm: An Experimental Evaluation and a Real-World Application

Peter Korošec, Jurij Šilc, Klemen Oblak, Franc Kosel

Abstract— **This paper describes the so-called Differential Ant-Stigmergy Algorithm (DASA), which is an extension of the Ant-Colony Optimization for a continuous domain. An experimental evaluation of the DASA on a benchmark suite from CEC 2005 is presented. The DASA is compared with a number of evolutionary optimization algorithms, including the covariance matrix adaptation evolutionary strategy, the differential evolution, the real-coded memetic algorithm, and the continuous estimation of distribution algorithm. The DASA is also compared to some other ant methods for continuous optimization. The experimental results demonstrate the promising performance of the new approach. Besides this experimental work, the DASA was applied to a real-world problem, where the efficiency of the radial impeller of a vacuum cleaner was optimized. As a result the aerodynamic power was increased by twenty per cent.**

I. INTRODUCTION

Real-parameter optimization is an important issue in many areas of human activities. The general problem is to find a set of parameter values, $\mathbf{x} = (x_1, x_2, \ldots, x_D)$, that minimizes a function, $f(\mathbf{x})$, of D real variables, i.e.,

$$\text{Find: } \mathbf{x}^* \mid f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^D.$$

In the past two or three decades, different kinds of optimization algorithms have been designed and applied to solve real-parameter function optimization problems. Some of the popular approaches are real-parameter genetic algorithms [20], evolution strategies [4], differential evolution [16], particle swarm optimization [9], classical methods such as quasi-Newton method [14], other non-evolutionary methods such as simulated annealing [10], tabu search [7] and lately ant-colony based algorithms.

Algorithms inspired by model of ant colony behavior are increasingly successful among researches in computer science and operational research. A particular successful metaheuristic—Ant Colony Optimization (ACO)—as a common framework for the existing applications and algorithmic variants of a variety of ant algorithms has been proposed by Dorigo and colleagues [5]. However, a direct application of the ACO for solving real-parameter optimization problem is difficult. The first algorithm designed for continuous function optimization was continuous ant colony optimization (CACO) [2] which comprises two levels: global and local. CACO uses the ant colony framework to perform local searches, whereas global search is handled by a genetic algorithm. Up to now, there are few other adaptations of ACO algorithm to continuous optimization problems: continuous interacting ant colony (CIAC) [6], ACO for continuous and mixed-variable (eACO) [15], and aggregation pheromone system [18].

Peter Korošec, peter.korosec@ijs.si (corresponding author), and Jurij Šilc, jurij.silc@ijs.si, are with the Jožef Stefan Institute, Ljubljana, Slovenia. Klemen Oblak, klemen.oblak@siol.net, is with the Domel Ltd., Železniki, Slovenia. Franc Kosel, franc.kosel@fs.uni-lj.si, is with the Faculty of Mechanical Engineering, University of Ljubljana, Slovenia.

In this paper we will introduce a new approach to the real-parameter optimization problem using an ACO-based algorithm that uses the pheromonal trail laying—a case of *stigmergy*—as a means of communication between ants.

The remainder of this paper is organized as follows. In Section II we introduce the optimization algorithm called the Differential Ant Stigmergy Algorithm. We round up with experimental evaluation on benchmark functions in Section III, followed by the application of the algorithm to the design of a radial impeller for vacuum cleaner in Section IV. Finally, we conclude the paper in Section V.

II. THE DIFFERENTIAL ANT STIGMERGY APPROACH

A. The Fine-Grained Discrete Form of Continuous Domain

In the following, a process of transformation from continuous domain into fine-grained discrete form is presented.

Let x'_i be a current value of the i-th parameter. During the searching for optimal parameter value, the new value, x_i, is assigned to the i-th parameter as follows:

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

Here, δ_i is a so-called *parameter difference* and is chosen from the set

$$\Delta_i = \Delta_i^- \cup \{0\} \cup \Delta_i^+,$$

where

$$\Delta_i^- = \{\delta_{i,k}^- \mid \delta_{i,k}^- = -b^{k+L_i-1}, k = 1, 2, \ldots, d_i\}$$

and

$$\Delta_i^+ = \{\delta_{i,k}^+ \mid \delta_{i,k}^+ = b^{k+L_i-1}, k = 1, 2, \ldots, d_i\}.$$

Here $d_i = U_i - L_i + 1$. Therefore, for each parameter x_i, parameter difference, δ_i, has a range from b^{L_i} to b^{U_i}, where b is so-called *discrete base*, $L_i = \lfloor \lg_b(\varepsilon_i) \rfloor$, and $U_i = \lfloor \lg_b(\max(x_i) - \min(x_i)) \rfloor$. With the parameter ε_i, the maximum precision of the parameter x_i is set. The precision is limited by the computer's floating-point arithmetics.

1-4244-1340-0/07$25.00 ©2007 IEEE

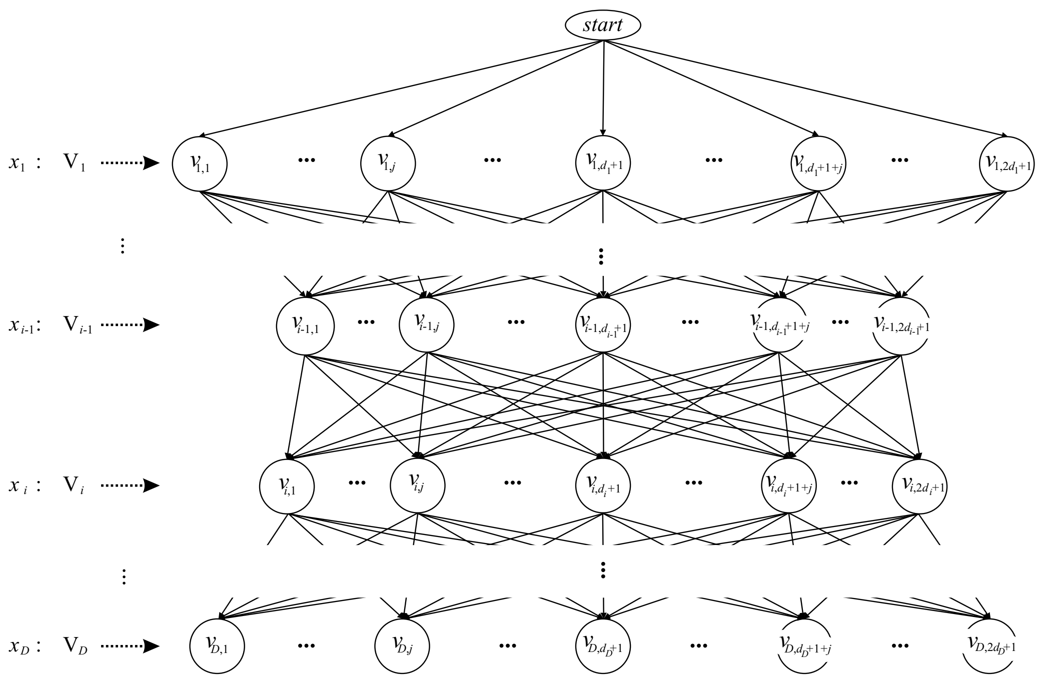


Fig. 1. Differential graph

B. Graph Representation

From all the sets Δ_i, $1 \leq i \leq D$, where D represents the number of parameters, a so-called *differential graph* $\mathcal{G} = (V, E)$ with a set of vertices, V, and a set of edges, E, between the vertices is constructed. Each set Δ_i is represented by the set of vertices, $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,2d_i+1}\}$, and $V = \bigcup_{i=1}^{D} V_i$. Then we have that

$$\Delta_i = \{\underbrace{\delta^-_{i,d_i}, \ldots, \delta^-_{i,d_i-j+1}, \ldots, \delta^-_{i,1}}_{\Delta_i^-}, 0, \underbrace{\delta^+_{i,1}, \ldots, \delta^+_{i,j}, \ldots, \delta^+_{i,d_i}}_{\Delta_i^+}\}$$

is equal to

$$V_i = \{v_{i,1}, \ldots, v_{i,j}, \ldots, \underbrace{v_{i,d_i+1}}_{0}, \ldots, v_{i,d_i+1+j}, \ldots, v_{i,2d_i+1}\},$$

where

$$v_{i,j} \xrightarrow{\delta} \delta^-_{i,d_i-(j-1)},$$
$$v_{i,d_i+1} \xrightarrow{\delta} 0,$$
$$v_{i,d_i+1+j} \xrightarrow{\delta} \delta^+_{i,j},$$

and $j = 1, 2, \ldots, d_i$. To enable a more flexible movement over the search space, the weight ω is added to Eq. 1:

$$x_i = x_i' + \omega\delta_i \tag{2}$$

where $\omega = \text{RandomInteger}(1, b-1)$.

Each vertex of the set V_i is connected to all the vertices that belong to the set V_{i+1} (see Fig. 1). Therefore, this is a directed graph, where each path ν from $start$ vertex to any of the ending vertices is of equal length and can be defined with v_i as:

$$\nu = (v_1 v_2 \ldots v_i \ldots v_D),$$

where $v_i \in V_i$, $1 \leq i \leq D$.

The optimization task is to find a path ν, such that $f(\mathbf{x}) < f(\mathbf{x}')$, where $\mathbf{x}'$ is currently the best solution, and $\mathbf{x} = \mathbf{x}' + \Delta(\nu)$ (using Eq. 1). Additionally, if the objective function $f(\mathbf{x})$ is smaller than $f(\mathbf{x}')$, then the $\mathbf{x}'$ values are replaced with $\mathbf{x}$ values.

C. The Differential Ant Stigmergy Algorithm (DASA)

The optimization consists of an iterative improvement of the currently best solution, $\mathbf{x}'$, by constructing an appropriate path ν, that uses Eq. 2 and returns a new best solution. This is done as follows:

1) A solution $\mathbf{x}'$ is manually set or randomly chosen.
2) A search graph is created and an initial amount of pheromone, $\tau^0_{V_i}$, is deposited on all the vertices from the set $V_i \subset V, 1 \leq i \leq D$, according to a Gaussian probability density function

$$\text{Gauss}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

where μ is the mean, σ is the standard deviation, and $\mu = 0$, $\sigma = 1$ (see Fig. 2).
3) There are m ants in a colony, all of which begin simultaneously from the $start$ vertex. Ants use a probability rule to determine which vertex will be chosen next. More specifically, ant α in step i moves from a vertex

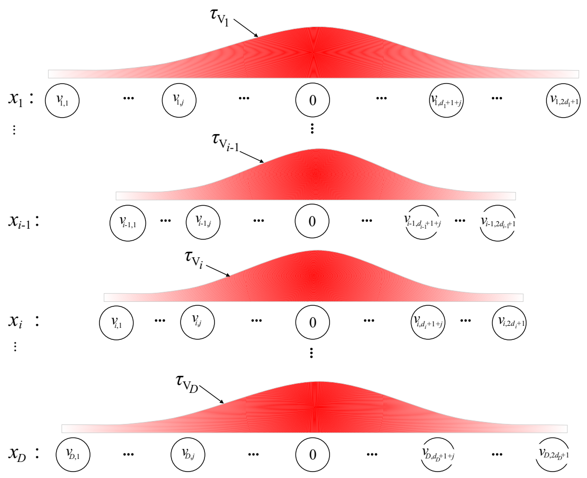


Fig. 2. Initial pheromone distribution

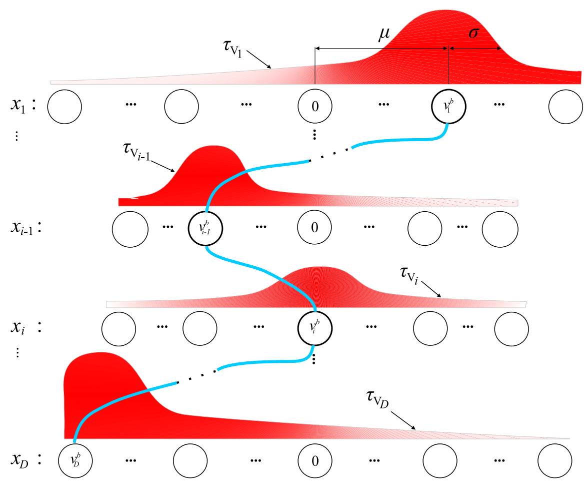


Fig. 3. Pheromone distribution after a new best solution is found

in set V_{i-1} to vertex $v_{i,j} \in \{v_{i,1}, \dots, v_{i,2d_i+1}\}$ with a probability given by:

$$p_j(\alpha, i) = \frac{\tau(v_{i,j})}{\sum_{1 \le k \le 2d_i+1} \tau(v_{i,k})},$$

where $\tau(v_{i,k})$ is the amount of pheromone on vertex $v_{i,k}$. The ants repeat this action until they reach the ending vertex. For each ant, solution $\mathbf{x}$ is constructed (see Eq. 2) and evaluated with a calculation of $f(\mathbf{x})$. The best solution, $\mathbf{x}^b$, out of m solutions is compared to the currently best solution $\mathbf{x}'$. If $f(\mathbf{x}^b)$ is better than $f(\mathbf{x}')$, then $\mathbf{x}'$ values are replaced with $\mathbf{x}^b$ values. Furthermore, in this case the pheromone amount is redistributed according to associated path $\nu^b = (v_1^b \dots, v_{i-1}^b v_i^b \dots v_D^b)$. New probability density functions have maxima on vertices v_i^b and the standard deviations are inversely proportioned to the improvements of the solutions (see Fig. 3).

4) Pheromone evaporation is defined by some predetermined percentage ρ on each probability density function as follows:

$$\mu^{\text{NEW}} = (1 - \rho)\mu^{\text{OLD}}$$

and

$$\sigma^{\text{NEW}} = \begin{cases} (1+\rho)\sigma^{\text{OLD}} & (1+\rho)\sigma^{\text{OLD}} < \sigma_{\max} \\ \sigma_{\max} & \text{otherwise} \end{cases}.$$

Pheromone dispersion has a similar effect as a pheromone evaporation in classical ACO algorithm.

5) The whole procedure is then repeated until some ending condition is met. Through the iterations of the algorithm we slowly decrease the maximum standard deviation, $\sigma_{\max}$, and with it improve the convergence (an example of daemon action).

We named the search algorithm presented in this section as *Differential Ant Stigmergy Algorithm* (DASA).

It is a well known that ant-based algorithms have problems with convergence. This happens when on each step of the walk there is a large number of possible vertices from which ant can choose from. But this is not the case with the DASA where Gaussian distribution of pheromone over each parameter was used. Namely, such distribution reduces the width of the graph to only few dominant parameter values (i.e., vertices). On the other hand, with proper selection of the discrete base, b, we can also improve the algorithm's convergence (larger b reduces the search graph size).

III. An Experimental Evaluation

A. The Experimental Environment

The computer platform used to perform the experiments was based on AMD Opteron™2.6-GHz processors, 2 GB of RAM, and the Microsoft®Windows®XP operating system.

The DASA has three parameters: the number of ants, m, the pheromone disperse factor, ρ, and the maximum parameter precision, ε. Their settings were: $m = 10$, $\rho = 0.1$, and $\varepsilon = 1\,\text{e–}12$. We must note that during the experimentation we did not fine-tune the algorithms parameters, but only make a limited number of experiments to find satisfying settings.

B. Test Benchmark Suite

The DASA was tested on four benchmark functions of dimension 30. The complete definition of the CEC 2005 test-suit is available in [17]. Function f_3 (*Shifted Rotated High Conditional Eliptic Function*) is unimodal and function f_9 (*Shifted Rastrigin's Function*) is multi-modal. Functions f_{13} (*Expanded Extended Griewank's plus Rosenbrock's Function*) and f_{15} (*Hybrid Composition Function*) result from the composition of several functions. To prevent exploitation of the symmetry of the search space and of the typical zero value associated with the global optimum, the local optimum is shifted to a value different from zero, and the function value of the global optimum is non-zero.

TABLE I

ALGORITHM COMPLEXITY (FUNCTION f_3, $D = 30$)

Algorithm	The system	Complexity
CMA-ES	Pentium 4 3GHz	$T_0 = 0.40$
	1GB RAM	$T_1 = 41.0$
	Red Hat Linux 2.4	$\widehat{T}_2 = 24.00^*$
	MATLAB 7.0.1	$\frac{\widehat{T}_2 - T_1}{T_0} = -$
DE	AMD Sempron 2800+	$T_0 = 0.29$
	1GB RAM	$T_1 = 7.64$
	Mandrake Linux 10.1	$\widehat{T}_2 = 8.49$
	C	$\frac{\widehat{T}_2 - T_1}{T_0} = 2.94$
MA	Pentium 4 2.8GHz	$T_0 = 0.42$
	512MB RAM	$T_1 = 8.63$
	Linux kernel v. 2.6	$\widehat{T}_2 = 13.45$
	C++ with GCC 3.3.2	$\frac{\widehat{T}_2 - T_1}{T_0} = 11.48$
EDA	Xeon 2.4GHz	$T_0 = 6.93^{**}$
	1GB RAM	$T_1 = 1.45$
	Windows XP (SP2)	$\widehat{T}_2 = 5.22$
	MATLAB 6	$\frac{\widehat{T}_2 - T_1}{T_0} = 0.54$
DASA	AMD Opteron 2.6GHz	$T_0 = 0.19$
	2GB RAM	$T_1 = 58.94$
	Windows XP (SP 2)	$\widehat{T}_2 = 59.20$
	Delphi	$\frac{\widehat{T}_2 - T_1}{T_0} = 1.37$

* The large number of T_1 reflect the large number of objective function calls, while for T_2 a complete, eventually large, population is evaluated (serially) within a single function call.

** Due to poor loop implementation in MATLAB 6.

C. Compared Algorithms

The DASA was compared to four well-known algorithms:

A restart *Covariance Matrix Adaptation Evolution Strategy* with increasing population size (CMA-ES) [1]: The CMA-ES introduced by Hansen and Ostermeier [8] is an evolutionary strategy that adapts the full covariance matrix of a normal search (mutation) distribution. By increasing the population size for each restart—as is suggested in [1]—the search characteristics become more global after each restart.

A *Differential Evolution* (DE) [13]: DE is a stochastic, population-based optimization algorithm. It was introduced by Storn and Price [16] and was developed to optimize the real (float) parameters of a real-valued function. DE resembles the structure of an evolutionary algorithm, but differs from traditional evolutionary algorithms in its generation of new candidate solutions and by its use of a "greedy" selection scheme.

A real-coded *Memetic Algorithm* (MA) [12]: The MA is genetic algorithm (GA) that apply a separate local search (LS) process to refine new individuals. The GA applied to make the exploration (i.e., to maintain diversity in population), the LS applied to improve new solutions (i.e., to exploit the most promising regions of the domain search). In [12] a steady-state GA is used.

A continuous *Estimation of Distribution Algorithm* (EDA) [21]: The EDA based on probabilistic modeling instead of classical genetic operators such as crossover or mutation. The EDA used in [21] employs a multivariate Gaussian distribution and is therefore able to represent correlation between variables in the selected individuals via the full covariance matrix of the system.

D. The Complexity of the Algorithm

To estimate the algorithm's complexity we have calculated $\frac{\widehat{T}_2 - T_1}{T_0}$, where computing time T_0 is independent of the function dimension and is calculated by running the benchmark algorithm below:

```
for i = 1 to 1,000,000
    x = (double) 5.55;
    x = x + x;
    x = x * x;
    x = sqrt(x);
    x = ln(x);
    x = exp(x);
    y = x/x
end
```

T_1 is the computing time for 200,000 evaluations only for function f_3 and $\widehat{T}_2$ is the mean time of five executions, but now considering the complete computing time of the algorithm for the function f_3. The results are included in Table I.

E. An Evaluation

The function error, $f(\mathbf{x}) - f(\mathbf{x}^*)$ being with $\mathbf{x}^*$ the optimum, is recorded after 300,000 function evaluations (FEs). The error value (EV) is collected for 25 runs after which the trials are ordered from best to worst. The trial mean and standard deviation as well as the results of best, median, and worst trial are presented in Table II.

Additionally, in Table III an average convergence after selected number of FEs is presented.

The results indicate the promising performance of the new approach. It is clear that our approach performs better than the rest of the approaches on three out of four test functions. Since the selected test functions reflect different kinds of pseudo-real optimization problems, one could expect that the DASA is applicable to many real-world multi-parameter optimization problems.

F. Comparison to Other Ant-Based Methods

As we mentioned in the introduction, there are few other adaptations of ACO algorithm to real-parameter optimization. Here, the DASA is compared to results presented by Socha in [15]. In order to have comparable results, the same accuracy level was chosen.

The results presented in Table IV are based on 25 independent runs of the DASA and show number of FEs to achieve the fixed accuracy level. The experimental results show that the DASA has much higher convergence speed than that of CACO and comparable with eACO.

TABLE II

EVs for thirty-dimensional f_3, f_9, f_{13} and f_{15}, measured after 300,000 FEs

EV	Algorithm				
	CMA-ES	DE	MA	EDA	DASA
Function f_3					
Best	4.07 e−9	5.46 e+4	5.55 e+5	2.27 e+6	1.27 e+5
Med	5.44 e−9	2.43 e+5	7.64 e+5	3.66 e+6	4.32 e+5
Worst	8.66 e−9	9.00 e+5	1.56 e+6	5.88 e+6	8.15 e+5
Mean	5.55 e−9	2.89 e+5	8.77 e+5	3.75 e+6	4.59 e+5
Std	1.09 e−9	1.93 e+5	5.81 e+4	9.09 e+5	2.02 e+5
Function f_9					
Best	4.35 e−6	0.00 e+0	7.78 e−9	2.10 e+2	0.00 e+0
Med	9.95 e−1	0.00 e+0	9.95 e−1	2.30 e+2	0.00 e+0
Worst	4.97 e+0	0.00 e+0	1.99 e+0	2.48 e+2	0.00 e+0
Mean	9.38 e−1	0.00 e+0	6.81 e−1	2.30 e+2	0.00 e+0
Std	1.18 e+0	0.00 e+0	1.21 e−1	9.44 e+0	0.00 e+0
Function f_{13}					
Best	1.10 e+0	2.31 e+0	1.33 e+0	3.82 e+1	9.62 e−1
Med	2.61 e+0	3.89 e+0	2.54 e+0	6.86 e+1	1.93 e+0
Worst	3.20 e+0	1.39 e+1	1.03 e+1	1.29 e+2	2.56 e+0
Mean	2.49 e+0	4.51 e+0	3.96 e+0	7.36 e+1	1.88 e+0
Std	5.13 e−1	2.26 e+0	5.38 e−1	2.36 e+1	3.99 e−1
Function f_{15}					
Best	2.00 e+2	4.75 e+2	2.00 e+2	4.35 e+2	0.00 e+0
Med	2.00 e+2	4.81 e+2	3.00 e+2	4.59 e+2	3.00 e+2
Worst	3.00 e+2	5.86 e+2	5.00 e+2	5.63 e+2	5.00 e+2
Mean	2.08 e+2	4.84 e+2	3.56 e+2	4.81 e+2	2.33 e+2
Std	2.75 e+1	2.14 e+1	1.51 e+1	4.67 e+1	1.58 e+2

TABLE III

Average EVs for thirty-dimensional f_3, f_9, f_{13} and f_{15}, measured after 1,000, 10,000, 100,000, and 300,000 FEs

FEs	Algorithm				
	CMA-ES	DE	MA	EDA	DASA
Function f_3					
1 e+3	1.07 e+9	5.53 e+8	2.94 e+8	1.25 e+9	3.10 e+8
1 e+4	6.11 e+6	8.15 e+7	4.14 e+7	2.76 e+8	1.16 e+7
1 e+5	5.55 e−9	1.52 e+6	5.51 e+6	3.49 e+7	1.23 e+6
3 e+5	5.55 e−9	2.89 e+5	8.77 e+5	3.75 e+6	4.59 e+5
Function f_9					
1 e+3	2.53 e+2	3.77 e+2	2.99 e+2	4.80 e+2	9.29 e+1
1 e+4	4.78 e+1	9.85 e+1	1.05 e+2	3.62 e+2	2.95 e+0
1 e+5	6.89 e+0	6.68 e−8	7.55 e+0	2.50 e+2	0.00 e+0
3 e+5	9.38 e−1	0.00 e+0	6.81 e−1	2.30 e+2	0.00 e+0
Function f_{13}					
1 e+3	1.14 e+2	1.62 e+5	3.95 e+3	7.50 e+5	2.12 e+5
1 e+4	3.80 e+0	1.02 e+2	1.51 e+1	3.08 e+5	7.02 e+0
1 e+5	2.89 e+0	4.55 e+0	8.66 e+0	4.52 e+3	2.04 e+0
3 e+5	2.49 e+0	4.51 e+0	3.96 e+0	7.36 e+1	1.88 e+0
Function f_{15}					
1 e+3	6.69 e+2	1.08 e+3	7.62 e+2	1.13 e+3	5.89 e+2
1 e+4	3.87 e+2	7.04 e+2	4.41 e+2	6.88 e+2	2.40 e+2
1 e+5	2.25 e+2	5.20 e+2	3.56 e+2	5.38 e+2	2.33 e+2
3 e+5	2.08 e+2	4.84 e+2	3.56 e+2	4.81 e+2	2.33 e+2

TABLE IV

Comparison of the average number of FEs until the accuracy is reached

Function*	Algorithm			
	CACO [2]	CIAC [6]	eACO [15]	DASA
Sphere	22,050	50,000	695	832
Goldstein & Price	5,320	23,391	364	991
Rosenbrock	6,842	11,797	2,905	137
Zakharov	—	—	401	182

* http://iridia.ulb.ac.be/∼ksocha/extaco04.html

Even though it was shown that the DASA performs well on benchmark functions, one always wonders how the algorithm will work on solving real-world applications. This question is answered in the next section.

IV. A Real-World Problem

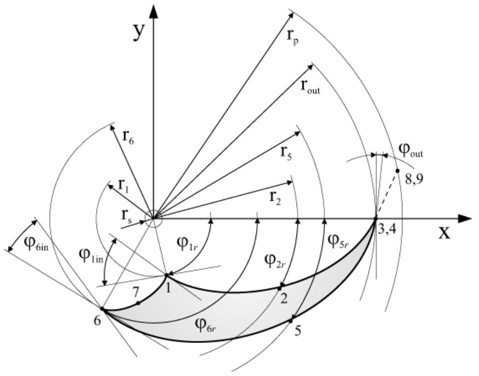
Besides the experimental work described in the previous section, we also applied the DASA to a real-world problem. Here, we optimized the radial impeller of a vacuum cleaner. Radial air impellers are the basic components of many turbomachines. In the following we will concentrate on relatively small impellers and subsonic speeds. Our main aim was to find an impeller shape that has a higher efficiency, i.e., greater aerodynamic power, than the one currently used in production.

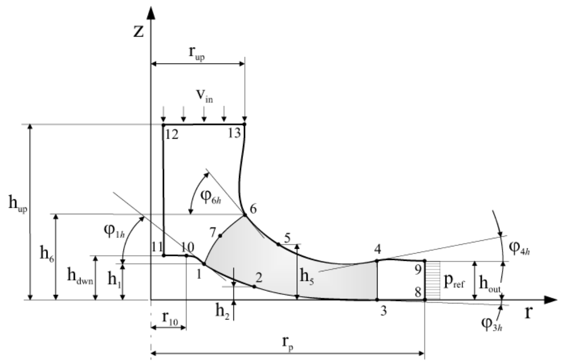
A. Modeling

An impeller is constructed from blades, an upper and a lower side. The sides enclose the blades and keep them together. The blades, which are all the same, were the main part of the optimization. The geometry of a blade is shown in Fig. 4, where the gray color represents the blade. The method of modeling is as follows: we construct the points at specific locations, draw the splines through them and spread the area on the splines. Once a blade is made an air channel must be constructed in a similar way.

In Fig. 4(a) the point 1 has two parameters: the radius r_1 and the angle $\varphi_{1\mathrm{r}}$. Similarly, the points 2, 5 and 6 have parameter pairs $r_2, \varphi_{2\mathrm{r}}$; $r_5, \varphi_{5\mathrm{r}}$ and $r_6, \varphi_{6\mathrm{r}}$. The points 3 and 4 are fixed on the x axis. This is because the impeller must have a constant outer radius, r_{out}, and the outer side of the blade must be parallel to the z axis. On the other hand, the outer angle of the blade, φ_{out}, and the angle of the spline at points 3 and 4, can be varied. Analogously, the angles $\varphi_{1\mathrm{in}}$ and $\varphi_{6\mathrm{in}}$ are the inner-blade angles for the upper and lower edges of the blade at the input, respectively.

In Fig. 4(b) the points 1, 2, and 3 form the upper spline, and the points 4, 5, and 6, the lower spline. Between the points 1 and 6 is the point 7, which defines the spline of the input shape of the blade. In this figure, the points 1, 2, 5,


(a)


(b)

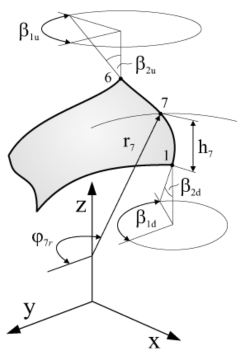

(c)

Fig. 4. *Parametric modeling: (a) top view; (b) side view; (c) 3D view*

and 6 have the parameters h_1, h_2, h_5, and h_6, respectively, describing their heights. Point 3 stays on the x axis and point 4 has a constant height, h_{out}. In other words, the designer of the impeller must know at least the outer diameter, r_{out}, and the height, h_{out}. The parameters φ_{1h} and φ_{6h} describe the input angles of the lower and upper parts of the blade with respect to the $r-z$ plane. Similarly, the parameters φ_{3h} and φ_{4h} describe the outer blade angle with respect to the same plane.

In Fig. 4(c) the meaning of point 7 is explained more precisely. The parameters r_7, h_7, and φ_{7r} define the radius, height, and angle, respectively. The radius and angle dictate where the point should appear with respect to the $x-y$ plane and the height with respect to the $r-z$ plane. Similarly, the angles $\beta_{\text{1u}}, \beta_{\text{2u}}, \beta_{\text{1d}}$, and β_{2d} are needed to define the starting and ending angles of the spline constructed between the points 1, 7, and 6.

If we look closely at Fig. 4(b) then we can see the contour surrounding the blade. This is the air channel with the following parameters: the inner radius, r_{s} (see Fig. 4(a)), which is needed for the hexahedral mesh (explained later), the air intake radius, r_{up}, the air outflow radius, r_{p}, the bolt radius, r_{10}, the bolt height, h_{dwn}, and the impeller height, h_{up}.

In this way we have successfully modeled the impeller geometry with 34 parameters. For each parameter we have a predefined search interval with a given discrete step. Therefore, the size of the search space can be obtained as the product of the number of possible settings over all the parameters. It turns out that there are approximately $3\,\text{e}{+}34$ possible solutions.

B. Estimation of Results

An example of the classical impeller (currently used in production) with nine blades and the corresponding air channel between the two blades are shown in Fig. 5. The mesh is constructed with more than 6,000 hexahedral elements. The boundary conditions are zero velocity at all the solid regions and symmetry boundary conditions at the fluid regions. At the influx and outflux (see Fig. 4(b)) the intake velocity, v_{in}, and reference pressure, p_{ref}, are defined, respectively. The intake velocity is parabolically distributed, because we expect that the intake flow is laminar and so:

$$v_{\text{in}} = v(\Phi(t))\frac{6r}{r_{\text{up}}}\left(\frac{r}{r_{\text{up}}} - 1\right).$$

Here, $v(\Phi(t))$ is a velocity dependent on the stream, which further depends on time, as we shall see later, r_{up} is the upper radius, defined before, and r is the radius within the limits from r_{s} to r_{up}. The reference pressure, p_{ref}, is set to zero.

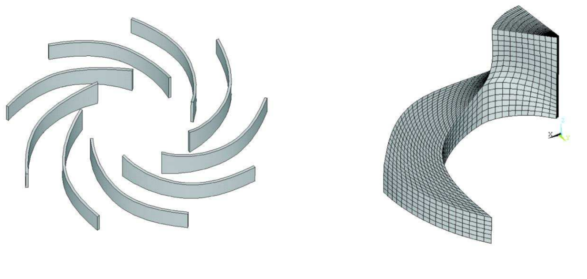


Fig. 5. *Geometry of the blades (left) and the hexahedral mesh of a periodic air channel (right)*

So far we have defined the geometry, the mesh and the boundary conditions. For the computational fluid dynamics (CFD) we will not give the theoretical background, which can be found in a lot of literature [3], [11], [19]. In our case, for the CFD we used the ANSYS®FLOTRAN$^{\text{TM}}$package. With respect to the maximum time, t_{max}, the flux is:

$$\Phi(t) = vA_{\text{in}}\frac{t}{t_{\text{max}}},$$

where A_{in} is the influx area and t is the current time. A relative pressure, as a result of the CFD calculation, is shown in Fig. 6.

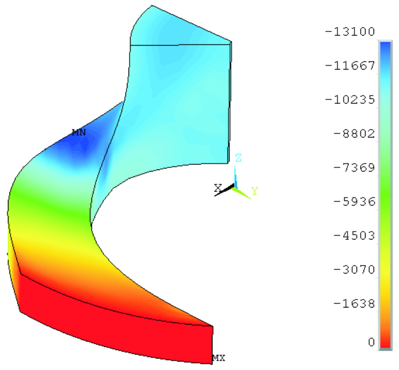


Fig. 6. *Relative pressure [Pa]*

The distribution of the relative pressure can be used to estimate the cost function. The average pressure, p_{in}, is chosen from the air-intake area. Finally, the aerodynamic power, W_{air}, which represents the cost function, is as follows:

$$W_{\text{air}} = (p_{\text{out}} - p_{\text{in}})\Phi(t_{\text{opt}}),$$

where p_{out} is the output pressure at the radius r_{out} and $\Phi(t_{\text{opt}}) = 40$ l/s is the flux near the desired optimum performance of the impeller. Our goal is to find such parameter-value settings, where W_{air} is maximized.

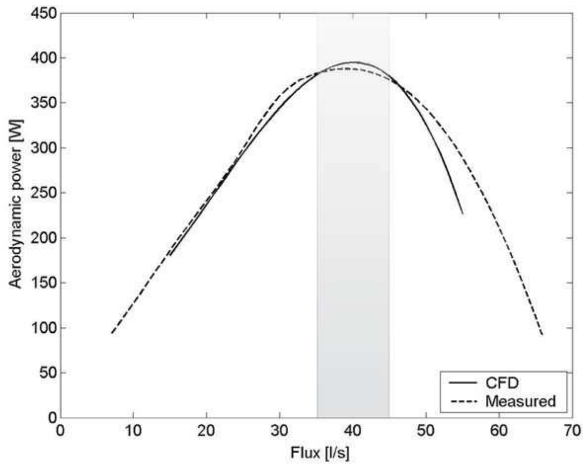


Fig. 7. *Measured and calculated aerodynamic power distribution of the classical impeller at $\omega = 40{,}000$ rpm*

Figure 7 shows the distribution of the aerodynamic power of the classical impeller. Here, we can see that the calculated values (solid line) of the aerodynamic power distribution closely match the measured values (dashed line). Both curves are interpolated from 10 points at different fluxes and a constant angular velocity $\omega = 40{,}000$ rpm. This result indicates that the selected CFD can be used for evaluation purposes.

C. Results

The optimization was performed on the same computer platform as the benchmark suite test in Section III. The DASA settings were: $m = 10$, $\rho = 0.2$, with ε dependent on the discrete step of each parameter.

The optimization method was run 10 times and each run consisted of 2,000 CFD calculations. A single CFD calculation takes approximately seven minutes. The obtained results, in terms of aerodynamic power, are presented statistically in Table V.

TABLE V
OPTIMIZED IMPELLER'S AERODYNAMIC POWER AFTER 2,000 CFD CLACULATIONS

$\Phi = 40$ l/s, $\omega = 40,000$ rpm	Classical impeller	Optimized impeller		
		Worst	Mean	Best
Aerodynamic power [W]	411.00	452.00	472.00	483.00

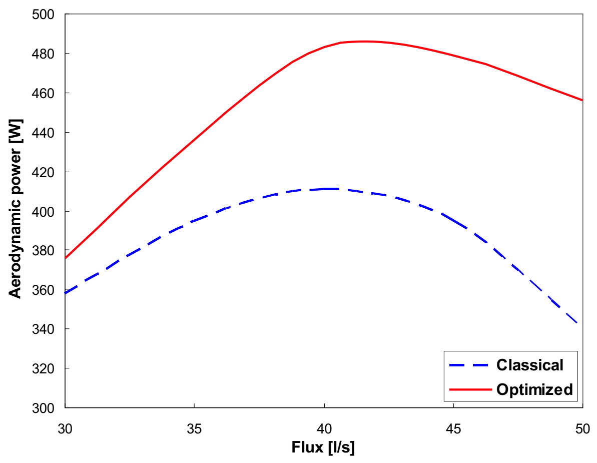


Fig. 8. *Aerodynamic power distribution of the classical and the optimized impeller at $\omega = 40{,}000$ rpm*

Figure 8 shows the aerodynamic power distribution of the classical and the optimized impeller (best of 10 runs) at $\omega = 40{,}000$ rpm. Here, we can see that at $\Phi(t_{\text{opt}}) = 40$ l/s the optimized impeller outperforms the classical one by 72 W.

Finally, Fig. 9 shows a 3D view of the optimized impeller.

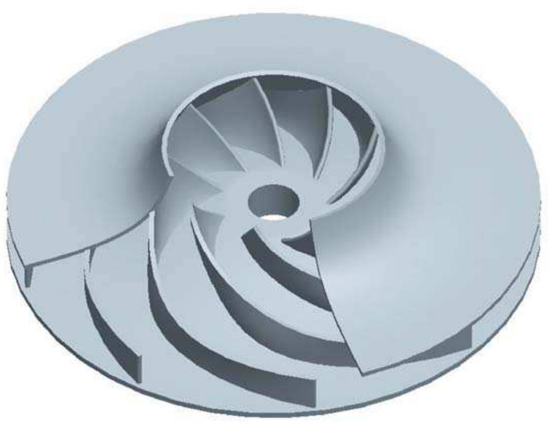


Fig. 9. *The optimized impeller*

V. Discussion and Conclusion

In this paper we introduced a new ACO-based metaheuristic called the Differential Ant-Stigmergy Algorithm (DASA) for continuous global optimization. As seen in Section II, the DASA is generally applicable to global optimization problems. In addition, it makes use of neither derivative nor a-priori information, making it an ideal solution method for *black-box* problems.

While it sometimes requires many function evaluations, Section III shows that for a selected set of CEC 2005 test functions, the DASA almost always converges to the global optimum in a reasonable time. Section IV shows the ability of the DASA to solve more challenging problems with real-world applications, thus making it a well-suited approach for solving global optimization problems from many fields of the physical sciences.

References

[1] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," *Proc. IEEE Congress on Evolutionary Computation*, Edinburgh, UK, Sept. 2005.

[2] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," *Lecture Notes in Coputer Science*, vol. 993, pp. 25–39, 1995.

[3] T. J. Chung, *Finite Element Analysis in Fluid Dynamics,* New York: McGraw-Hill Education, 1978.

[4] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evolutionary Computation*, vol. 10, pp. 371–395, Winter 2002.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization,* Cambridge, Massachusetts: The MIT Press, 2004.

[6] J. Dréo and P. Siarry, "A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions," *Lecture Notes in Coputer Science*, vol. 2463, pp. 216–227, 2002.

[7] F. Glover and M. Laguna, *Tabu Search,* Boston: Kluwer Academic Publishers, 1997.

[8] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distribution in evolutionary strategies: The covariance matrix adaptation," *Proc. IEEE Conference on Evolutionary Computation*, Nagoya, Japan, May 1996, pp. 312–317.

[9] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proc. IEEE International Conference on Neural Networks*, Perth, Australia, Nov./Dec. 1995, pp. 1942-1948.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] H. Lomax, T. H. Pulliam, and D. W., Zingg, *Fundamentals of Computational Fluid Dynamics,* Berlin: Springer-Verlag 2001.

[12] D. Molina, F. Herrera, and M. Lozano, "Adaptive local search parameters for real-coded memetic algorithms," *Proc. IEEE Congress on Evolutionary Computation*, Edinburgh, UK, Sept. 2005.

[13] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," *Proc. IEEE Congress on Evolutionary Computation*, Edinburgh, UK, Sept. 2005.

[14] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell, *Engineering Optimization Methods*, New York: Wiley, 1983.

[15] K. Socha, "ACO for continuous and mixed-variable optimization," *Lecture Notes in Coputer Science*, vol. 3172, pp. 25–36, 2004.

[16] R. Storn and K. V. Price, "Differential evolution – A simple and efficient huristic for global optimization over continuous space," *Journal of Global Optimization*, vol. 11, pp. 341–359, Dec. 1997.

[17] P. N. Sunganthan, N. Hansen, J. J. Liang, Y. -P. Chen, A. Auger, and S. Tiwari, *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*, Technical Report, Nanyang Technological University, Singapore, May 2005. `http://www.ntu.edu.sg/home/EPNSugan`

[18] S. Tsutsui, "An enhanced aggregation pheromone system for real-parameter optimization in the ACO metaphor," *Lecture Notes in Coputer Science*, vol. 4150, pp. 60–71, 2006.

[19] H. K. Versteeg and W., Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method Approach,* Harlow, Essex: Longman Scientific & Technical, 1995.

[20] A. H. Wright, "Genetic algorithms for real parameter optimization," *Proc. 1st Workshop on Foundations of Genetic Algorithms*, Bloomington, IN, July 1990, pp. 205–218.

[21] B. Yuan and M. Gallagher, "Experimental results for the special session on real-parameter optimization at CEC 2005: A simple, continuous EDA," *Proc. IEEE Congress on Evolutionary Computation*, Edinburgh, UK, Sept. 2005.