We are IntechOpen, the world’s leading publisher of Open Access books
Built by scientists, for scientists

6,500 Open access books available
175,000 International authors and editors
190M Downloads

154 Countries delivered to
TOP 1% Our authors are among the most cited scientists
12.2% Contributors from top 500 universities

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com
Applications of the Differential Ant-Stigmergy Algorithm on Real-World Continuous Optimization Problems

Peter Korošec and Jurij Šilc
Jožef Stefan Institute, Ljubljana
Slovenia

1. Introduction

Ants have always fascinated human beings. What particularly strikes the occasional observer as well as the scientist is the high degree of societal organization that these insects can achieve in spite of very limited individual capabilities (Dorigo et al., 2000). Ants have inspired also a number of optimization algorithms. These algorithms are increasingly successful among researchers in computer science and operational research (Blum, 2005; Cordón et al., 2002; Dorigo & Stützle, 2004).

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 in the early nineties by Marco Dorigo (Dorigo, 1992). ACO takes inspiration from the foraging behavior of some ant species. These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving combinatorial optimization problems.

In recent years ACO algorithms have been applied to more challenging and complex problem domains. One such domain is continuous optimization. However, a direct application of the ACO for solving continuous optimization problem is difficult. The first algorithm designed for continuous function optimization was continuous ant colony optimization (Bilchev & Parmee, 1995) which comprises two levels: global and local; it 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 (Dréo & Siarry, 2002), ACO for continuous and mixed-variable (Socha, 2004), aggregation pheromone system (Tsutsui, 2006), and multilevel ant-stigmergy algorithm (Korošec & Šilc, 2008).

In this chapter we will present so-called Differential Ant-Stigmergy Algorithm (DASA), a new approach to the continuous optimization problem (Korošec, 2006). We start with the DASA description followed by three case studies which show real-world application of the proposed optimization approach. Finally, we conclude with discussion of the obtained results.
Fig. 1. Stigmergy is an organizing principle in emergent systems in which the individual parts of the system communicate with one another indirectly by modifying their local environment. Ant colonies are a classic example. The ants communicate indirectly. Information is exchanged through modifications of the environment (local gradients of pheromone).

2. A differential ant-stigmergy approach

2.1 Continuous optimization
The general continuous optimization problem is to find a set of parameter values, \( p = (p_1, p_2, \ldots, p_D) \), that minimizes a function, \( f_{\text{cost}}(p) \), of \( D \) real variables, i.e.,

\[
\text{Find: } \quad p^* \quad | \quad f_{\text{cost}}(p^*) \leq f_{\text{cost}}(p), \forall p \in \mathbb{R}^D.
\]

To solve this problem, we created a fine-grained discrete form of continuous domain. With it we were able to represent this problem as a graph. This enabled us to use ant-based approach for solving numerical optimization problems.

2.2 The fine-grained discrete form of continuous domain
Let \( p'_i \) be the current value of the \( i \)-th parameter. During the searching for the optimal parameter value, the new value, \( p_i \), is assigned to the \( i \)-th parameter as follows:

\[
p_i = p'_i + \delta_i.
\]

Here, \( \delta_i \) is the so-called parameter difference and is chosen from the set
\[ \Delta_i = \Delta_i^+ \cup \emptyset \cup \Delta_i^- \]  
(3)

where

\[ \Delta_i^- = \{ \delta_{i,k}^+ | \delta_{i,k}^- = -b^{k-1}, k = 1,2, \ldots d_i \} \]  
(4)

and

\[ \Delta_i^+ = \{ \delta_{i,k}^- | \delta_{i,k}^+ = b^{k-1}, k = 1,2, \ldots d_i \} \]  
(5)

Here

\[ d_i = U_i - L_i + 1. \]  
(6)

Therefore, for each parameter \( p_i \), the parameter difference, \( \delta_i \), has a range from \( b^{L_i} \) to \( b^{U_i} \), where \( b \) is the so-called discrete base,

\[ L_i = \left \lfloor \log_b (p_i) \right \rfloor \]  
(7)

and

\[ U_i = \left \lfloor \log_b (\max(p_i) - \min(p_i)) \right \rfloor \]  
(8)

With the parameter \( \delta_i \), the maximum precision of the parameter \( p_i \) is set. The precision is limited by the computer's floating-point arithmetic. To enable a more flexible movement over the search space, the weight \( \omega \) is added to Eq. 2:

\[ p_i = p_i^j + \omega \delta_i. \]  
(9)

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

### 2.3 Graph representation

From all the sets \( \Delta_i \), \( 1 \leq i \leq D \), where \( D \) represents the number of parameters, a so-called search graph \( G = (V, E) \) with a set of vertices, \( V \), and a set of edges, \( E \), between the vertices is constructed (see Fig. 2). Each set \( \Delta_i \) is represented by the set of vertices,

\[ V_i = \{ v_{i,1}, v_{i,2}, \ldots, v_{i,2d_i+1} \}, \]  
(10)

and

\[ V = \bigcup_{i=1}^{D} V_i. \]  
(11)

Then we have that

\[ \Delta_i = \{ \delta_{i,0}, \delta_{i,1}, \ldots, \delta_{i,j-1}, \delta_{i,j}, \delta_{i,j+1}, \ldots, \delta_{i,d_i} \} \]  
(12)

is equal to

\[ V_i = \{ v_{i,1}, v_{i,2}, \ldots, v_{i,j-1}, v_{i,j+1}, \ldots, v_{i,2d_i+1} \}, \]  
(12)

where \( j = 1,2, \ldots, d_i \).
Each vertex of the set $V_i$ is connected to all the vertices that belong to the set $V_{i+1}$. Therefore, this is a directed graph, where each path $\nu$ from start vertex to any of the ending vertices is of equal length and can be defined with $v_i$ as:

$$\nu = (v_1, v_2, ..., v_D),$$

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

The optimization task is to find a path $\nu$, such that $f_{cost}(p) \leq f_{cost}(p')$, where $p'$ is currently the best solution, and $p = p' + \Delta(\nu)$ (using Eq. 9). Additionally, if the objective function $f_{cost}(p)$ is smaller than $f_{cost}(p')$, then the $p'$ values are replaced with $p$ values.

### 2.4 The differential ant stigmergy algorithm

The optimization consists of an iterative improvement of the currently best solution, $p'$, by constructing an appropriate path $\nu$, that uses Eq. 9 and returns a new best solution. This is done as follows (see Fig. 3):
Fig. 3. The Differential Ant-Stigmergy Algorithm (DASA)

**Step 1.** A solution \( p \) is manually set or randomly chosen.

**Step 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 the Cauchy probability density function.
\[ C(x) = \frac{1}{s \pi + \frac{\pi}{8} (x - l)^2}, \quad (14) \]

where \( l \) is the location offset and \( s = s_{\text{global}} - s_{\text{local}} \) is the scale factor. For an initial pheromone distribution the standard Cauchy distribution \((l = 0, \ s_{\text{global}} = 1, \ \text{and}\ s_{\text{local}} = 0)\) is used and each parameter vertices are equidistantly arranged between \( z = [-4, 4] \).

**Step 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 \( \alpha \) in step \( i \) moves from a vertex in set \( V_i \) to vertex \( v_{i,j} \in \{v_{i,1}, v_{i,2}, \ldots, v_{i,2d+1}\} \) with a probability, \( \text{prob} \), given by:

\[
\text{prob}(\alpha, i) = \frac{\tau(v_{i,j})}{\sum_{1 \leq k \leq 2d+1} \tau(v_{i,k})}, \quad (15)
\]

where \( \tau(v_{i,j}) \) is the amount of pheromone on vertex \( v_{i,j} \).

The ants repeat this action until they reach the ending vertex. For each ant, path \( \nu \) is constructed. If for some predetermined number of tries we get \( \nu = 0 \) the search process is reset by randomly choosing new \( p_{\text{temporary_best}} \) and pheromone re-initialization. New solution \( p \) is constructed (see Eq. 9) and evaluated with a calculation of \( f_{\text{cost}}(p) \). The current best solution, \( p_{\text{current_best}} \), out of \( m \) solutions is compared to the temporary best solution \( p_{\text{temporary_best}} \). If \( p_{\text{current_best}} \) is better than \( p_{\text{temporary_best}} \), then \( p_{\text{temporary_best}} \) values are replaced with \( p_{\text{current_best}} \) values. In this case \( s_{\text{global}} \) is increased (in our case for 1 %) and pheromone amount is redistributed according to the associated path \( \nu_{\text{best}} \). Furthermore, if new \( p_{\text{temporary_best}} \) is better then \( p_{\text{best}} \), then \( p_{\text{best}} \) values are replaced with \( p_{\text{temporary_best}} \) values. So, global best solution is stored. If no better solution is found \( s_{\text{local}} \) is decreased (in our case for 3 %).

**Step 4.** Pheromone dispersion is defined by some predetermined percentage \( \chi \). The probability density function \( C(x) \) is changed in the following way:

\[
l \leftarrow (1 - \chi)l, \quad (16)
\]

and

\[
s_{\text{local}} \leftarrow (1 - \chi)s_{\text{local}}. \quad (17)
\]

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

**Step 5.** The whole procedure is then repeated until some ending condition is met. 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 Cauchy 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).
2.5 Software implementation and parameter settings
The DASA was implemented in the Borland® Delphi™ programming language (see Fig. 4). The computer platform used to perform the optimizations was based on AMD Opteron™ 2.6-GHz processors, 2 GB of RAM, and the Microsoft® Windows® XP 32-bit operating system.

The DASA parameters were set as follows: the number of ants, \( m \), was set to 10, the pheromone evaporation factor, \( \chi \), was set to 0.2, and the maximum parameter precision, \( \varepsilon \), dependent on the discrete step of each parameter.

![Fig. 4. GUI used for setting the DASA options](image)

3. Case studies
Case studies presented in this section are related to the development of a new dry vacuum cleaner (Korošec et al., 2007; Tušar et al., 2007). In particular, the efficiency of the turbo-compressor unit (see Fig. 5) was improved.

3.1 An electric motor power losses minimization
Home appliances, such as vacuum cleaners and mixers, are generally powered by a universal electric motor (UM). These appliances need as low as energy consumption, that is, input power, as possible, while still satisfying the needs of the user by providing sufficient output power. The optimization task is to find the geometrical parameter values that will generate the rotor and the stator geometries resulting in the minimum power losses.
Fig. 5. Turbo-compressor unit of a dry vacuum cleaner

Fig. 6. Parameters that define the rotor and stator geometries

There are several invariable and variable parameters that define the rotor and stator geometries. The invariable parameters are fixed; they cannot be altered, either for technical reasons (e.g., the air gap) or because of the physical constraints on the motor (e.g., the radius of the rotor's shaft). The variable parameters do not have predefined optimum values. Some variable parameters are mutually independent and without any constraints. Others are dependent, either on some invariable parameters or on mutually independent ones.

In our case, 11 mutually independent variable parameters defining the rotor and stator geometries are subject to optimization (see Fig. 6): rotor yoke thickness, $p_1$, rotor external radius, $p_2$, rotor pole width, $p_3$, stator yoke horizontal thickness, $p_4$, stator yoke vertical thickness, $p_5$, stator middle-part length, $p_6$, stator internal edge radius, $p_7$, stator teeth radius, $p_8$, stator slot radius, $p_9$, stator width, $p_{10}$, and stator height, $p_{11}$. We optimized only 10 parameters (the ratio between $p_{10}$ and $p_{11}$ was set constant) of the UM rotor and stator geometries.
Power losses, $P_{\text{losses}}$, are the main factor affecting the efficiency of a UM. The efficiency of a UM, $\eta$, is defined as the ratio of the output power, $P_{\text{out}}$, to the input power, $P_{\text{inp}}$:

$$\eta = \frac{P_{\text{out}}}{P_{\text{inp}}} = \frac{P_{\text{out}}}{P_{\text{out}} + P_{\text{losses}}}$$

(18)

and it is very dependent on various power losses (Sen, 1997):

$$P_{\text{losses}} = P_{\text{Cu}} + P_{\text{Fe}} + P_{\text{brush}} + P_{\text{vent}} + P_{\text{frict}}.$$  

(19)

The overall copper losses, $P_{\text{Cu}}$, occurring in the rotor and the stator slots are as follows:

$$P_{\text{Cu}} = \sum_i (J A \rho l_{\text{turn}}),$$

(20)

where $i$ stands for each slot, $J$ is the current density, $A$ is the slot area, $\rho$ is the copper's specific resistance and $l_{\text{turn}}$ is the length of the winding turn. The calculation of the iron losses, $P_{\text{Fe}}$, is less exact because iron has non-linear magnetic characteristics. The iron losses are therefore separated into two components: the hysteresis losses and the eddy-current losses. This means the motor's iron losses can be expressed with the formula:

$$P_{\text{Fe}} = c_e B^2 f_e^2 m_r + c_h B^2 f_e^2 m_s + c_b B^2 f_e m_s,$$

(21)

where $c_e$ is the eddy-current material constant at 50Hz, $c_h$ is the hysteresis material constant at 50Hz, $B$ is the maximum magnetic flux density, $f_e$ is the frequency of the magnetic field density in the rotor, $f_s$ is the frequency of the magnetic field density in the stator, $m_r$ is the mass of the rotor, and $m_s$ is the mass of the stator. Three additional types of losses occurring in the UM, i.e., the brush losses, $P_{\text{brush}}$; the ventilation losses, $P_{\text{vent}}$; and the friction losses, $P_{\text{frict}}$, depend mainly on the speed of the motor. When optimizing the geometries of the rotor and the stator, the motor's speed is assumed to be fixed; hence $P_{\text{brush}}$, $P_{\text{vent}}$, and $P_{\text{frict}}$ have no impact on the motor's efficiency, and so these losses are not significantly affected by the geometries of the rotor and the stator. Therefore, in our case, the overall copper and iron losses can be used to estimate the cost function:

$$f_{c}(p) = P_{\text{Cu}} + P_{\text{Fe}}.$$  

(22)

Our goal is to find such parameter-value settings, where $f_{c}(p)$ is minimized.

The DASA starts its search with the existent solution. The stopping criterion was set to 1,400 calculations. The calculation of a single solution via the ANSYS Multiphysics simulation takes approximately two minutes, which means that the execution of 1,400 calculations needs about two days. The optimization method was run 20 times. The obtained results in terms of the UM power losses are presented statistically in Table 1.

<table>
<thead>
<tr>
<th>Existing solution</th>
<th>Optimized solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>Mean</td>
</tr>
<tr>
<td>177.9</td>
<td>136.7</td>
</tr>
</tbody>
</table>

Table 1. Optimized UM's power losses in Watts after 1,400 calculations
The engineering rotor and stator design (the existent solution) results in power losses of 177.9 W, and can be seen in Fig. 7 (left). The figure shows the magnetic flux density in the laminations (higher magnetic flux density, $B$, causes higher power losses). Figure 7 (right) present a typical example of a feasible rotor and stator geometry, with power losses of 129.1 W. This solution has very low iron losses in the rotor due to its small size and in spite of its high magnetic saturation. The small rotor and its saturation are compensated by large stator poles that ensure large enough a magnetic flux. This design is completely feasible from the technical and production points of view.

3.2 An electric motor casing stiffness maximization

The casing is a part of the dry vacuum cleaner motor which is built into vacuum cleaners. The casing is basically an axisymmetric shell structure which is built of steel suitable for forming. For this procedure which consists of eleven different phases it is important that the radii are growing or do not change while the height is growing. That is an important rule which must not be broken during the geometry optimization. The goal of optimization is to preserve the stiffness while using a thinner shell structure and consequently save material and reduce costs.

The computational model of the casing comprises 26 different parameters which generate the geometry variations. The classical parameters are: radius on the top, $p_1$, radius on the side, $p_2$, radius at the groove for brushes, $p_3$, deviation of the hole for diffuser fixation, $p_4$, radius of roundness at the beginning of the bearing groove, $p_5$, radius on the top of the bearing groove, $p_6$, radius on the top of air culverts, $p_7$, radius at the bottom of air culverts, $p_8$, roundness of air culverts, $p_9$, angle of slope, $p_{10}$, angle of culverts span, $p_{11}$, slope of bearing groove, $p_{12}$, height of bearing groove, $p_{13}$, and slope of the brushes groove, $p_{14}$.

Besides the above listed 14 classical parameters there are 12 more ($p_{15}$ to $p_{26}$), with which the ribs on areas A, B and C are defined (Fig. 8). These are essential to improve the stiffness so we will search for those which maximize the cost function. With these parameters we can generate a lot of combinations of different rib shapes, depending on the number of given paths and accuracy of intervals for points deviations, positions and number of ribs. In our
case there were several billion combinations for ribs. It is obvious that instead of a defined path at position \( j \), we could define displacements for individual points but this would result in many more parameters and time consuming calculation procedure. The question still remains what would be the benefit of that. From previous designs there exist some reasonable shapes for ribs which have been defined intuitively and these then change their position, magnitude and number, which are probably sufficient for a good result, as there has been over ten different rib shapes defined earlier.

Fig. 8. Parameters of a casing and ribs. Ribs A (top), B (center), and C (bottom); basic (left) and modify forms (center and right).

Besides the parameters for the creation of ribs there are other classical parameters which also have an influence on their shape. Namely, if the radii \( p_1 \) or \( p_2 \) are varying then this is reflected in the shape of ribs of model A or if the height of bearing groove, \( p_{25r} \), and slope of groove, \( p_{24r} \), are modified then this will be reflected in the ribs of model C, etc.

The cost function is relatively simply defined when loadings are deterministic. Usually we optimize such cases with the minimization of displacements or stresses, maximization of elasticity, etc. (Dražušerčič & Kosel, 2005). It is difficult to determine the cost function for a dynamically loaded assembly where the loads are stochastic. So the question is how to define the stiffness of a shell structure if we do not know the lateral loads. Let us consider a certain arbitrarily shaped plane shell. We can write its potential energy, \( W_p \), as:

\[
W_p = \frac{1}{2} \int \sigma_{ij} \varepsilon_{ij} \, dV.
\]

If we consider that if the Poisson’s shear modulus is neglected, the stress tensor, \( \sigma_{ij} \), and strain tensor, \( \varepsilon_{ij} \), are the following:

\[
\sigma_{ij} = -2 \begin{pmatrix}
\frac{\partial^2 w(x,y)}{\partial x^2}, \frac{\partial^2 w(x,y)}{\partial y^2}, \frac{\partial^2 w(x,y)}{\partial x \partial y}
\end{pmatrix}^T
\]

and

www.intechopen.com
\[ \varepsilon_u = -2E \left\{ \frac{\partial^2 w(x,y)}{\partial x^2}, \frac{\partial^2 w(x,y)}{\partial y^2}, \frac{\partial^2 w(x,y)}{\partial x \partial y} \right\}^T. \] (25)

Here \( w(x,y) \) is displacement in \( z \) direction and \( E \) is elasticity modulus. The potential energy can then be written as:

\[ W_p = \frac{1}{2E} \int \left( \sigma_{xx}^2 + \sigma_{yy}^2 + 2 \tau_{xy}^2 \right) \, dV. \] (26)

We can see that in this case the potential energy is an integral of squares of stresses over the volume. We define also the kinetic energy, \( W_k \), for plate vibrations:

\[ W_k = \frac{\rho \omega^2}{2} \int \omega^2(x,y) \, dV, \] (27)

where \( \rho \) represents the density of material. Here the second derivative of displacement is approximated with respect to time with the product of squared displacement and eigen frequency, \( \omega \).

For conservative systems the maximal potential energy is equal to maximal kinetic energy and from this follows the expression for cost function which can be in our case the stiffness itself:

\[ f_c(p) = \frac{\int \left( \sigma_{xx}^2 + \sigma_{yy}^2 + 2 \tau_{xy}^2 \right) \, dV}{\int \omega^2(x,y) \, dV}. \] (28)

The DASA was run 20 times and each run consisted of 2,000 calculations. The cost function was calculated by the ANSYS Multiphysics simulation tool. The obtained results are presented statistically in Table 2.

<table>
<thead>
<tr>
<th>Existing solution</th>
<th>Optimized solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Worst</td>
</tr>
<tr>
<td>5.87 · 10^{-3}</td>
<td>6.13 · 10^{-3}</td>
</tr>
</tbody>
</table>

Table 2. Optimized casing’s stiffness after 2,000 CFD calculations

The results of optimization were quite surprising (Fig. 9 bottom row). It was expected that the ribs would form on all the given surface but they did not. The ribs were more distinct where the vertical part of the surface was turning into the horizontal one (ribs A). There were no ribs at the surface where the stator is in a tight fit, probably because of pre-stressing of the casing through the stator. Also in the groove for brushes (ribs B) there were no distinct ribs, probably because the groove itself is a kind of rib. Instead of this there is a given slope for a groove which was not there before. The radii of roundings were in most cases bigger than before as we can clearly see at the air culverts (Fig. 9 middle bottom).
3.3 A turbo-compressor aerodynamic power maximization

Radial air impellers are the basic components of many turbo-machines. In the following we will concentrate on relatively small impellers and subsonic speeds used in a dry vacuum cleaner. 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.

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. 10, 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. 10a the point 1 has two parameters: the radius \( p_1 \) and the angle \( p_2 \). Similarly, the points 2, 5 and 6 have parameter pairs \( p_3 \) and \( p_4 \), \( p_5 \) and \( p_6 \), \( p_7 \) and \( p_8 \). The points 3 and 4 are fixed on the \( x \) axis. This is because the impeller must have a constant outer radius \( p_9 \) and the outer side of the blade must be parallel to the \( z \) axis. On the other hand, the outer angle of the blade \( p_{10} \) and the angle of the spline at points 3 and 4, can be varied. Analogously, the angles \( p_{11} \) and \( p_{12} \) are the inner-blade angles for the upper and lower edges of the blade at the input, respectively.

In Fig. 10c 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, and 6 have the parameters \( p_{13} \), \( p_{14} \), \( p_{15} \), and \( p_{16} \), respectively, describing their heights.
Point 3 stays on the $x$ axis and point 4 has a constant height $p_{17}$. In other words, the designer of the impeller must know at least the outer diameter $p_9$ and the height $p_{17}$. The parameters $p_{18}$ and $p_{19}$ describe the input angles of the lower and upper parts of the blade with respect to the $r-z$ plane. Similarly, the parameters $p_{20}$ and $p_{21}$ describe the outer blade angle with respect to the same plane.

In Fig. 10b the meaning of point 7 is explained more precisely. The parameters $p_{22}$, $p_{23}$, and $p_{24}$ 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 $p_{25}$, $p_{26}$, $p_{27}$, and $p_{28}$ 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. 10c then we can see the contour surrounding the blade. This is the air channel with the following parameters: the inner radius $p_{29}$, the air intake radius $p_{30}$, the air outflow radius $p_{31}$, the bolt radius $p_{32}$, the bolt height $p_{33}$, and the impeller height $p_{34}$. 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 $3 \times 10^4$ possible solutions.

The mesh of the air channel between the two blades 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 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:

$$v_{in} = v(\Phi(t)) \frac{6r}{r_{up}} \left( \frac{r}{r_{up}} - 1 \right),$$  \hspace{1cm} (29)$$

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. With respect to the maximum time, $t_{max}$, the flux is:

$$\Phi(t) = v A_{in} \frac{t}{t_{max}},$$  \hspace{1cm} (30)$$

where $A_{in}$ is the influx area and $t$ is the current time.

Fig. 11. The existent impeller (left), the worst optimized impeller (middle), and the best optimized impeller (right)

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, which represents the cost function, is as follows:

$$f_c(p) = P_{air} = (p_{out} - p_{in}) \Phi(t_{opt}),$$  \hspace{1cm} (31)$$

www.intechopen.com
where $p_{\text{out}}$ is the output pressure at the radius $r_{\text{out}}$ and $\Phi(t_{\text{opt}}) = 401/s$ is the flux near the desired optimum performance of the impeller. Our goal is to find such parameter-value settings, where $P_{\text{opt}}$ is maximized.

The DASA was run 10 times and each run consisted of 2,000 CFD calculations. For the CFD calculations we used the ANSYS Multiphysics package. A single CFD calculation takes approximately seven minutes. The obtained results, in terms of aerodynamic power, are presented statistically in Table 3.

<table>
<thead>
<tr>
<th>Existing solution</th>
<th>Optimized solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>Mean</td>
</tr>
<tr>
<td>411.00</td>
<td>432.00</td>
</tr>
</tbody>
</table>

Table 3 Optimized impeller's aerodynamic power in watts after 2,000 CFD calculations

Results show that we were able to increase aerodynamic power for approximately 20 %. Figure 11 shows a 3D view of the existent and two optimized impellers (best and worst of 10 runs).

4. Discussion

In this chapter the Differential Ant-Stigmergy Algorithm (DASA) was presented, where the main goal was an evaluation of the DASA on some real-world engineering applications. Case studies were selected from a R&D project where more efficient turbo-compressor for dry vacuum cleaner was developed. Here the DASA was used to improve the efficiency of an electric motor, increase casing stiffness, and increase impeller's aerodynamic power. In all these cases the improvement was evident.

![Fig. 12. The Bonferroni-Dunn post-hoc statistical test](www.intechopen.com)
In (Korošec & Šilc, 2009a) we also compared the DASA to the eleven state-of-the-art stochastic algorithms for continuous optimization which were presented in the CEC’2005 Special Session on Real Parameter Optimization. The algorithms are:

- BLX-GL50, a hybrid real-coded genetic algorithm with female and male differentiation (García-Martínez & Lozano, 2005),
- BLX-MA, a real-coded memetic algorithm with adaptive local search parameters (Molina et al., 2005),
- CoEVO, an evolutionary algorithm with mutation step co-evolution (Pošik, 2005),
- DE, a differential evolution (Rönkkönen et al., 2005),
- EDA, a continuous estimation of distribution algorithm (Yuan & Gallagher, 2005),
- FEA, a flexible evolutionary algorithm (Alonso et al., 2005),
- G-CMA-ES, a restart covariance matrix adaptation evolutionary strategy with increasing population size (Auger & Hansen, 2005a),
- G-DE, a guided differential evolution (Bui et al., 2005),
- K-PCX, a population-based steady-state procedure (Sinha et al., 2005),
- L-CMA-ES, an advanced local search evolutionary algorithm (Auger & Hansen, 2005b), and
- SPC-PNX, a steady-state real-parameter genetic algorithm (Ballester et al., 2005).

Obtained results confirmed that the DASA is comparable to above algorithms and therefore generally applicable to global optimization problems. The experimental results have shown that the DASA ranks among the best algorithms for real-life-like applications. Its main advantage is in consistent problem solving which is indicated by 19 rankings in top third, 4 ranking in middle third and only 2 in bottom third. Statistical analysis following the Bonferroni-Dunn post-hoc test with $\alpha = 0.10$ showed that the DASA is significantly better than 8 out of 11 compared algorithms.

The algorithm was also applied to dynamic optimization problems with continuous variables proposed for CEC’2009 Special Session on Evolutionary Computation in Dynamic and Uncertain Environments (Korošec & Šilc, 2009b). If we compare the DASA to:

- a self-adaptive differential evolution (Brest et al., 2009),
- a dynamic artificial immune algorithm (de França & Von Zuben, 2009),
- an evolutionary programming (Yu & Suganthan, 2009), and
- a clustering particle swarm optimizer (Li & Yang, 2009),

the results show that the DASA can find reasonable solutions for all of the problems. It can be seen that the DASA performs not many worse than a self-adaptive differential evolution and much better than the other three algorithms. 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. 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.

5. References


Applications of the Differential Ant-Stigmergy Algorithm on Real-World Continuous Optimization Problems


www.intechopen.com

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:
