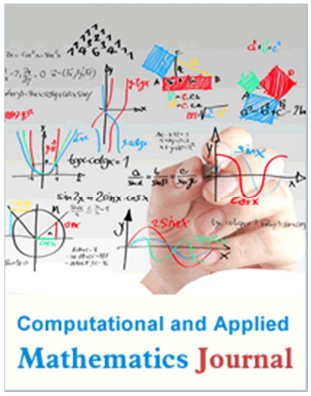
Computational and Applied Mathematics Journal
2015; 1(5): 307-318
Published online July 10, 2015 (http://www.aascit.org/journal/camj)



Keywords
Imperialist Competitive Algorithm,
Nelder-Mead,
Hybrid Algorithm,
Electric Motor Design

Received: June 8, 2015
Revised: June 22, 2015
Accepted: June 23, 2015

Hybrid Nelder-Mead Imperialist Competitive Algorithm Applied to Electric Motor Design

Julien Lepagnot[1], Lhassane Idoumghar[1], Daniel Fodorean[2]

[1]LMIA Laboratory (EA 3993), University of Haute-Alsace, Mulhouse, France
[2]Electrical Machines & Drives Department, Technical University of Cluj-Napoca, Cluj-Napoca, Romania

E-mail address
julien.lepagnot@uha.fr (J. Lepagnot), lhassane.idoumghar@uha.fr (L. Idoumghar)

Citation
Julien Lepagnot, Lhassane Idoumghar, Daniel Fodorean. Hybrid Nelder-Mead Imperialist Competitive Algorithm Applied to Electric Motor Design. *Computational and Applied Mathematics Journal.* Vol. 1, No. 5, 2015, pp. 307-318.

Abstract
In this paper, a hybrid metaheuristic based on Imperialist Competitive Algorithm (ICA) and on the Nelder-Mead simplex method (NM) is proposed. The purpose of NM is to improve both the diversification and the intensification capabilities of ICA. The proposed algorithm, called ICA-NM, is validated and compared with ICA and two other well-known metaheuristics using the benchmark of the CEC'2005 congress. The results show the efficiency of the proposed algorithm. Then, ICA-NM is used to design an optimal motor for an electric scooter in terms of both its mass and its output power. For this practical problem, ICA-NM outperforms several well-known metaheuristics used for this problem. Finally, the solution found by ICA-NM is validated by building and testing the corresponding prototype motor.

1. Introduction

Imperialist competitive algorithm (ICA) [1] is a population based metaheuristic in which there are two types of individuals, called countries: colonies and imperialists that form together a set of empires. ICA is based on imperialistic competition between these empires. During this competition, weak empires collapse and powerful ones take possession of their colonies until only one empire remains.

We propose to hybridize ICA with the *Nelder-Mead simplex* [2] in order to improve its performances. We have selected this simplex search method because it is easy to program, fast and widely used. Hybridizing NM with metaheuristics like evolutionary algorithms, particle swarm optimization and ant colony optimization has been a very popular approach to improve their intensification capabilities [3, 4, 5, 6]. In this paper, NM is also used to improve the diversification capabilities of ICA. As soon as a stagnation criterion is satisfied for ICA, NM is run in order to either escape the reached local optimum or speed up the convergence to it.

The proposed hybrid algorithm, called ICA-NM, is evaluated using the benchmark of the 2005 IEEE Congress on Evolutionary Computation (CEC2005) special session on real-parameter optimization [7]. This experimental analysis shows the performance of ICA-NM for different kinds of problems. It shows also that ICA-NM obtains good performances compared to other well-known metaheuristics. Then, it is successfully used to design a permanent-magnet machine used to motorize an electric scooter. The solutions found by ICA-NM for this practical problem are better than the ones obtained by ICA and other leading approaches used for this problem. This paper is structured as follows: Section 2 presents an overview of ICA and NM. The proposed ICA-NM algorithm is then

described in detail in Section 3. Experimental protocol and parameter setting are presented in Section 4. Experimental results are discussed in Section 5. Finally, a conclusion is given in Section 6.

2. Presentation of the Hybridized Components

2.1. Overview of Imperialist Competitive Algorithm

Imperialist Competitive Algorithm (ICA) [1] is a recent evolutionary optimization approach inspired by imperialism and the imperialistic competition process. In this algorithm, all individuals are grouped in several empires. The mechanisms in the algorithm are designed to bring out an empire, stronger than the others, and that finds the best solutions. Imperialistic competition aims to destroy the weakest empire and strengthen the strongest empire. The main steps of the algorithm are summarized in Algorithm 1.

Algorithm 1. *Imperialist Competitive Algorithm.*

1: Initialize and evaluate the empires
2: **while** stop condition is not satisfied **do**
3: Move the colonies toward their relevant imperialist
4: **if** a colony in an empire has a lower cost than the imperialist **then**
5: Switch the positions of that colony and of the imperialist
6: **end if**
7: Compute the total cost of all empires
8: **if** the distance between two empires is less than *Uniting Threshold* **then**
9: Merge the two empires
10: **end if**
11: Imperialistic competition
12: **if** there is an empire with no colony **then**
13: Destroy this empire
14: **end if**
15: **end while**

2.1.1. Initial Empires

Like all evolutionary algorithms, ICA starts with an initial population of solutions, called countries, of size N_{pop}. From these countries, N_{imp} best solutions are selected to be imperialists and the remaining N_{col} countries form the colonies of these imperialists. The initial empires are formed by dividing the colonies among imperialists according to their normalized power:

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \tag{1}$$

where $C_n = c_n - \max_i c_i$, c_n is the cost of n^{th} imperialist and C_n is its normalized cost.

The number of colonies NC_n that form an empire is computed according to (2):

$$NC_n = round(p_n \cdot N_{col}) \tag{2}$$

where the *round* function rounds a number to the nearest integer.

2.1.2. Movement of Each Colony

Imperialist countries attract colonies toward themselves using the assimilation policy illustrated in Fig. 1. To update its position, each colony moves toward its imperialist using (3):

$$x_{t+1} = x_t + \beta \cdot d \cdot \gamma \cdot U(-\theta, \theta) \tag{3}$$

where t is an iteration number, $\beta > 1$ causes the colonies to get closer to the imperialist, d is the distance between the colony and its imperialist, γ is a small number that corresponds to an assimilation coefficient ($0 < \gamma < 1$), and θ is a parameter that adjusts the deviation from the original direction, which enables searching around the imperialist.

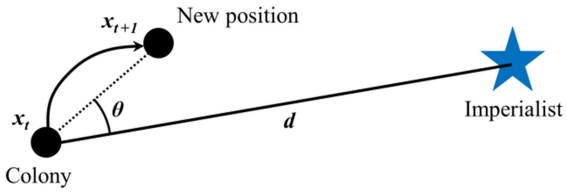


Fig. 1. *The movement of a colony toward its imperialist.*

2.1.3. Total Cost

The total cost of each empire is defined by the cost of its imperialist plus its average colonies' cost, as expressed by (4):

$$\begin{aligned} TC_n = \; & Cost(Imperialist_n) \\ & + \xi \cdot mean(Cost(coloniesofempire_n)) \end{aligned} \tag{4}$$

where ξ is a positive number considered to be less than 1.

2.1.4. Imperialistic Competition

All empires try to take possession of the colonies of other empires and to control them. This imperialistic competition is modeled by picking the weakest colony(ies) from the weakest empires and giving it (them) to the empire that has the highest likelihood to possess it (them). To start the competition, the possession probability P_{p_n} of each empire must be defined as:

$$P_{p_n} = \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \tag{5}$$

where $NTC_n = TC_n - \max_i TC_i$, TC_n and NTC_n are the total cost and the normalized total cost of n^{th} empire respectively.

To divide the mentioned colonies among empires based on their possession probability, a vector P is formed:

$$P = [P_{p_1}, P_{p_2}, P_{p_3}, ..., P_{p_{N_{imp}}}] \tag{6}$$

Then a vector with the same size as P, filled with random numbers, is created:

$$R = [r_1, r_2, r_3, ..., r_{N_{imp}}] \tag{7}$$

where r_i are random numbers between 0 and 1. Finally, a vector A is formed by subtracting R from P.

$$A = [P_{p_1} - r_1, P_{p_2} - r_2, ..., P_{p_{N_{imp}}} - r_{N_{imp}}] \tag{8}$$

Referring to vector A, the mentioned colonies will be assigned to an empire whose relevant index in A is maximum.

2.2. The Nelder-Mead Simplex

The Nelder-Mead simplex search method (NM), first proposed by Spendley et al. [8] and later refined by Nelder and Mead [2], is a derivative-free line search method that was particularly designed for traditional unconstrained minimization scenarios, such as the problems of nonlinear least squares and nonlinear simultaneous equations.

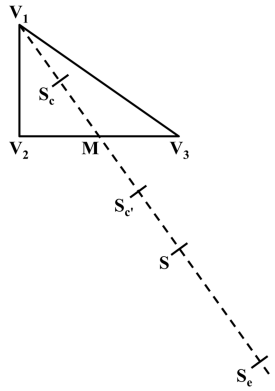


Fig. 2. *Illustration of the Nelder-Mead simplex algorithm. Either the worst point* V_1 *is replaced with* S, S_e, S_c *or* $S_{c'}$, *or the simplex is shrinked.*

A simplex is a polytope of $n+1$ vertices in a n-dimensional space. The algorithm starts with an initial simplex and evolves by using four elementary geometric transformations: reflection, expansion, contraction and shrinkage. Through these operations, the simplex can improve itself and come closer and closer to a local optimum. An example of the function minimization of two variables, taken from [3], is given to illustrate the basic procedure of NM. The function to minimize is denoted by f. Firstly, the method starts with an initial simplex designed with the points (solutions) denoted by V_1, V_2 and V_3, as illustrated in Fig.

2. Suppose $f(V_1)$ is the highest (the worst) of the three function values. Then, V_1 has to be replaced. In this case, a reflection is made through the centroid of the other points (the midpoint M) to the point S. The reflected solution is given by (9).

$$S = M + (M - V_1) \tag{9}$$

Suppose $f(V_3) < f(V_2) < f(V_1)$. At this stage, three situations can arise:

1. If $f(S) < f(V_3)$, an extension is made to the point S_e according to (10), where χ is called the expansion coefficient and it has a standard value of 2. Then, we keep S or S_e as a replacement for V_1, depending on which function value is lower.

$$S_e = M + \chi(M - V_1) \tag{10}$$

2. If $f(S) > f(V_3)$, a contraction is made to the point S_c or $S_{c'}$ according to (11) and (12), depending on whether $f(V_1)$ or $f(S)$ is lower. In these equations, ρ is called the contraction coefficient and it has a standard value of 0.5.

$$S_c = M - \rho(M - V_1) \tag{11}$$

$$S_{c'} = M + \rho(M - V_1) \tag{12}$$

3. If $f(S_c)$ or $f(S_{c'})$ is greater than $f(V_3)$, then the contraction has failed and we perform a shrinkage operation. The shrinkage operation reduces the size of the simplex by moving all but the best point V_3 halfway towards V_3.

3. The Proposed Hybrid ICA-NM Algorithm

The goal of combining ICA and NM is to take advantage of strengths from both methods. NM is a very efficient local search procedure. Hence, it can be used to improve the ICA intensification capabilities. Besides, the local optimum found by NM depends on the starting points selected to form the initial simplex. Then, NM could also be used to improve the diversification capabilities of ICA. Indeed, by using a proper set of points to initialize NM, it can converge to a solution located in a promising or unexplored area of the search space.

From these considerations, we propose to run several iterations of NM as soon as a stagnation criterion is satisfied. The number of iterations performed is denoted by N_{it}, and $N_{it} = c_{it} \times dim$ where c_{it} is a predefined coefficient and dim is the number of dimensions of the search space. The criterion used to trigger an execution of NM is satisfied if the ICA algorithm is showing signs of a possible premature convergence. Then, the use of NM can guide ICA towards new promising areas of the search space.

Each time NM is run, a new initial simplex is used. This

initial simplex is a set of $dim+1$ points in the search space. We denote this set by V, and the i^{th} point of this set by V_i. The set V is computed from another set of $dim+1$ points, denoted by U. We propose to use the $dim+1$ best ICA imperialists as the set of points U. If there is not enough imperialists to form U, *i.e.* if $N_{imp} < dim+1$ where N_{imp} is the number of imperialists, then the missing $(dim+1)-N_{imp}$ points are randomly generated according to a uniform distribution in the search space. The computation of the initial simplex V from the set of points U is described in Algorithm 2, where c_{spx} is a variable of ICA-NM initialized to $c_{spx}=1$ at the beginning of ICA-NM. The variable c_{spx} is decreased after each run of NM (line 6 of Algorithm 3), in order to initialize the initial simplex of NM with vertices that are closer and closer to the best imperialist.

Algorithm 2. *Computation of the initial simplex of NM.*

1: $bi \leftarrow$ index of the best imperialist in U
2: $\mathrm{V}_{bi} \leftarrow \mathrm{U}_{bi}$
3: **for all** $\mathrm{U}_i \in \mathrm{U}$ such that $i \neq bi$ **do**
4: $\quad \mathrm{V}_i \leftarrow \mathrm{U}_{bi} + c_{spx}\left(\mathrm{U}_i - \mathrm{U}_{bi}\right)$
5: **end for**

The stagnation criterion used to trigger an execution of NM is satisfied if no improvement of the best imperialist (replacement with a better colony) is observed during δ_t iterations (not necessarily successive ones) of ICA, where δ_t is a predefined threshold. The execution of NM takes place at the end of an ICA iteration, *i.e.* after the 14^{th} line of Algorithm 1. The pseudocode inserted at this place in Algorithm 1 is presented in Algorithm 3, where $count_i$ is the number of non-improving iterations of the i^{th} imperialist and c_{damp} is a predefined coefficient (a strictly positive real value lower or equal to 1).

Algorithm 3. *Integration of NM into ICA*

1: $best \leftarrow$ index of the empire of the best imperialist
2: **if** $count_{best} > \delta_t$ **then**
3: $\quad count_{best} \leftarrow 0$
4: $\quad$ Perform N_{it} iterations of NM
5: $\quad N_{it} \leftarrow \mathrm{round}\left(\dfrac{N_{it}}{c_{damp}}\right)$
6: $\quad c_{spx} \leftarrow c_{spx} \times c_{damp}$
7: **end if**

If the solution found by NM is better than the best imperialist, then it replaces the best imperialist. This way, the use of NM can improve not only the diversification capabilities of ICA, but also its intensification capabilities by fine-tuning the best imperialist.

4. Experimental Protocol and Parameter Setting

4.1. The CEC 2005 Benchmark

In order to evaluate the performance of ICA-NM, nine commonly used benchmark functions from the CEC 2005 special session are employed [7]. These functions are shifted and rotated variants of the classical mathematical functions in order to provide difficult test cases. The objective is to find their global minimum, using a maximum number of evaluations of a function equal to $10000 \times dim$, where dim is the dimension of the function. The performance of the algorithm is evaluated using $dim = 50$ and $dim = 100$. Table 1 lists the aforementioned functions respectively, where *Range* is the boundary of the function's search space, and f_{min} is the value of the global minimum of the function. Their detailed description is available in the CEC 2005 technical report [7].

Table 1. *Benchmark functions.*

Name	Description	Range	f_{min}
F_1	Shifted Sphere	[-100, 100]	-450
F_2	Shifted Schwefel's Problem 1.2	[-100, 100]	-450
F_3	Shifted Rotated High Conditioned Elliptic	[-100, 100]	-450
F_4	Schwefel's Problem 2.6 (Optimum on Bounds)	[-100, 100]	-310
F_5	Shifted Rosenbrock's	[-100, 100]	390
F_6	Shifted Rastrigin's	[-5, 5]	-330
F_7	Shifted Rotated Rastrigin's	[-5, 5]	-330
F_8	Shifted Rotated Weierstrass	[-0.5, 0.5]	90
F_9	Schwefel's Problem 2.13	$[-\pi, \pi]$	-460

4.2. Defining the Application

The autonomy of an electric vehicle (EV), or electric scooter, is reduced by the heavy components installed on board. Consequently, if we reduce the mass of the equipment, we will increase the EV's autonomy.

The main elements from an electric traction chain of an EV are: the battery, the converter and the electric motor. The battery volume and weight are imposed by its capacity to store

energy and of the used material: lead-acid (heavier, but cheaper), lithium-ion (lighter, but more expensive). The converter has several electronic devices which do not weight much. The main element in the traction system which can be reduced in its weight, by that reducing its cost too, is the electric motor. Thus, our goal is to have a lighter motorization with decreased investment.

The motorization of an electric vehicle can be realized with mechanical transmission or without it. The mechanical transmission supposes that the electric motor is placed on the chassis and the torque is transmitted to the wheels via a belt, gear or a distribution system.

The other variant is to use a non-transmission motorization, meaning that the electric motor is mounted within the wheel of the EV. For the electric scooter, in order to spare space and to reduce its weight, it is better to place the motor inside the wheel (without mechanical transmission). In such case, the motor has the rotor outside (it rotates with the tire), and the stator armature is inside. Such a motor is called *in-wheel motor*.

Fig. 3 depicts the power flow of the traction chain for the studied experimental scooter. Here, we can distinguish:

- the battery (which assures the necessary input energy) ;
- the converter (a 3-phase inverter in this case, used to transfer the energy from the battery to the electric motor) ;
- the in-wheel motor (which will produce the necessary torque and speed to run the scooter).

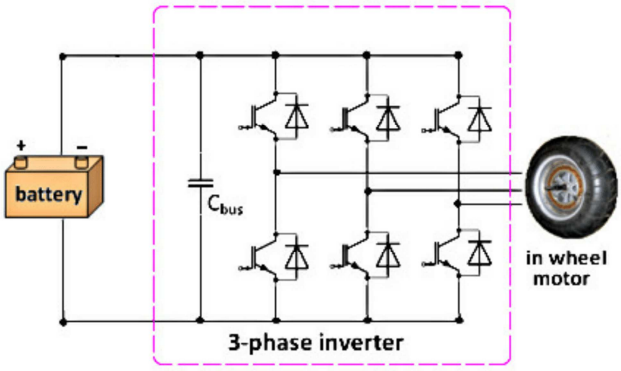
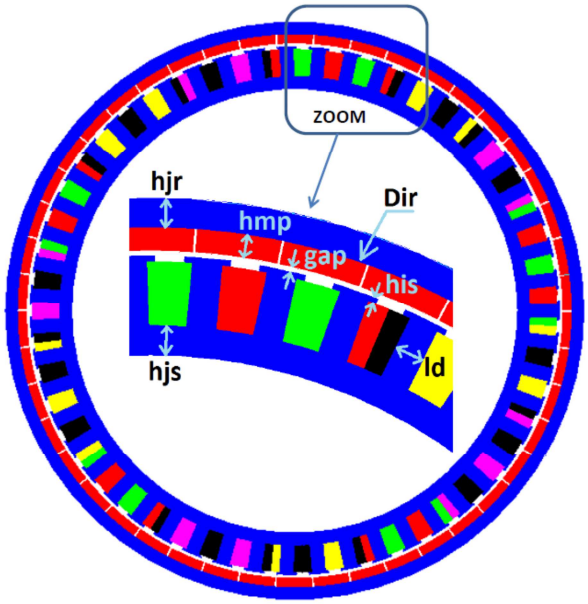


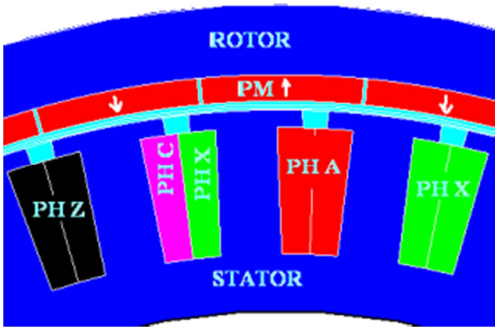
Fig. 3. *The electrical circuit for the traction of an electric scooter.*

The authors' goal is to reduce, as much as possible, the mass of the electric motor. The main design data of the electric scooter motorization is: 1200W for the power, 48Vdc for the battery, 34Nm for the rated torque, 420 r/min for the rated speed (corresponds to the scooter maximum speed, 50 km/h).

Related to the used motorization, the studied motor has an outer-rotor, excited with permanent magnets, while the stator is inside; thus, the motor is called outer rotor permanent magnet synchronous machine (OR-PMSM). The output performances expected at the motor's shaft are: 1.5 kW for the power, 420 r/min for the rated speed, meaning that the torque will be 34.1 Nm. After the calculations, the phase rated current of the motor will be 21 A. For in-wheel motors, it is very important to have a very compact volume for the machine, meaning a reduced mass. This is achieved with a higher number of poles, which also offers the possibility to employ a specific winding (of fractional slot type) which will produce a very smooth torque [9]. In this case, the number of poles pair is 17 and the three phase winding is installed in 39 slots. The reader can have an idea about the machine's configuration by looking at Fig. 4. Here, one can see also the geometrical parameters which will be used in the optimization process. The initial value and the limits of these parameters are presented in Table 2, and they will establish the intervals which will define the space of available solutions in which the global optimum will be searched.



(a)



(b)

Fig. 4. *The studied OR-PMSM: (a) cross section and geometrical parameters used in the optimization process ; (b) the active parts of the motor: PMs, rotor and stator iron.*

The optimization approach will try to maximize the power density of the machine (i.e., the ratio between the output power and the mass of the active parts of the machine). Since we do not have the interest of increasing the power (which is demanded by the application), the optimization problem given here consists in minimizing the mass of the machine. The components of the active parts of the OR-PMSM which we are trying to minimize are: the PMs, the rotor and stator iron.

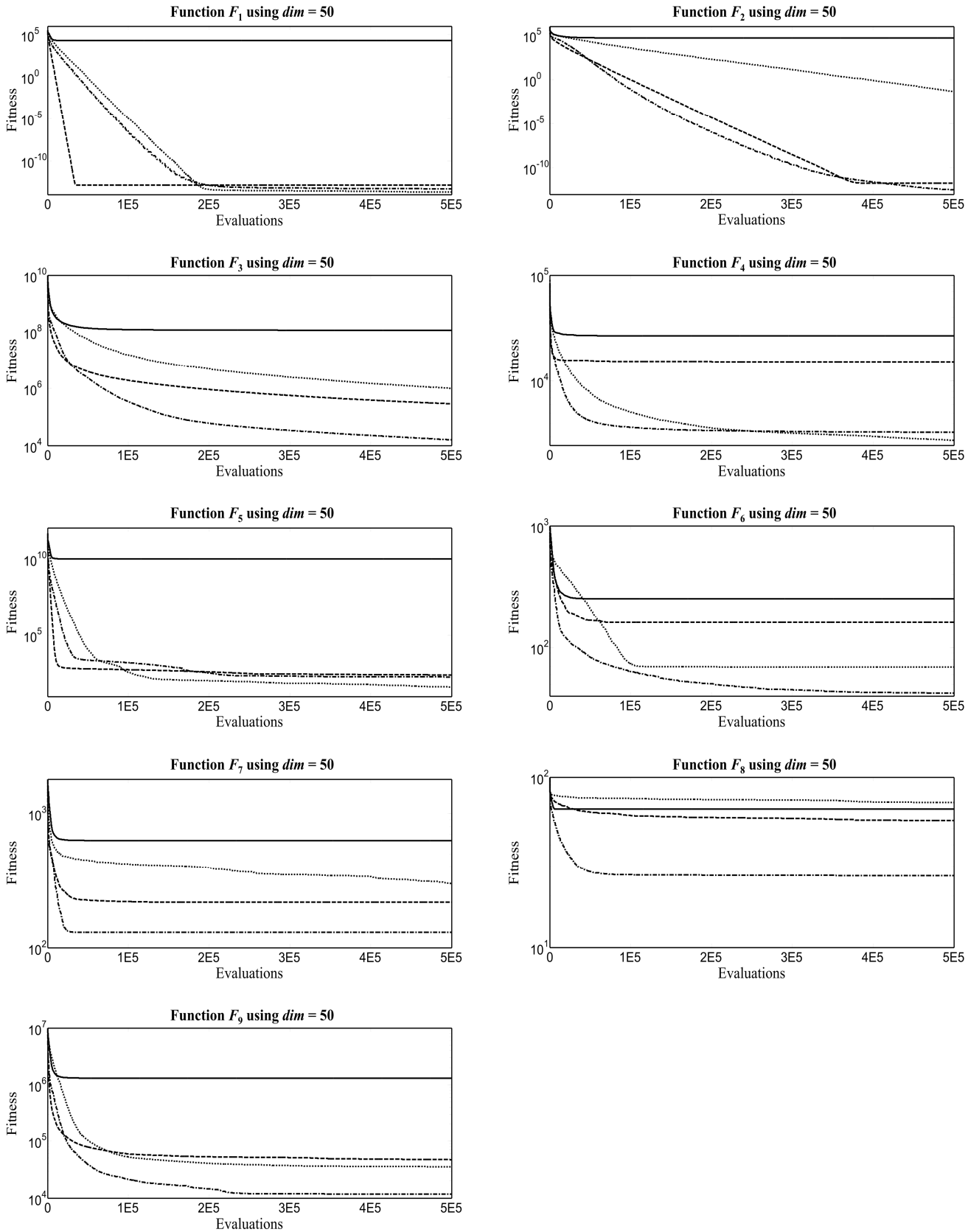


Fig. 5. *Evolution of the average value of the best solution found by ICA-NM, ICA, SPSO2011 and DE for the CEC 2005 benchmark functions using* $dim = 50$ *. A solid line is used for ICA, a dashed line is used for SPSO2011, a dotted line is used for DE and a dash-dot line is used for ICA-NM. For more clarity, we used a logarithmic scale for the ordinates.*

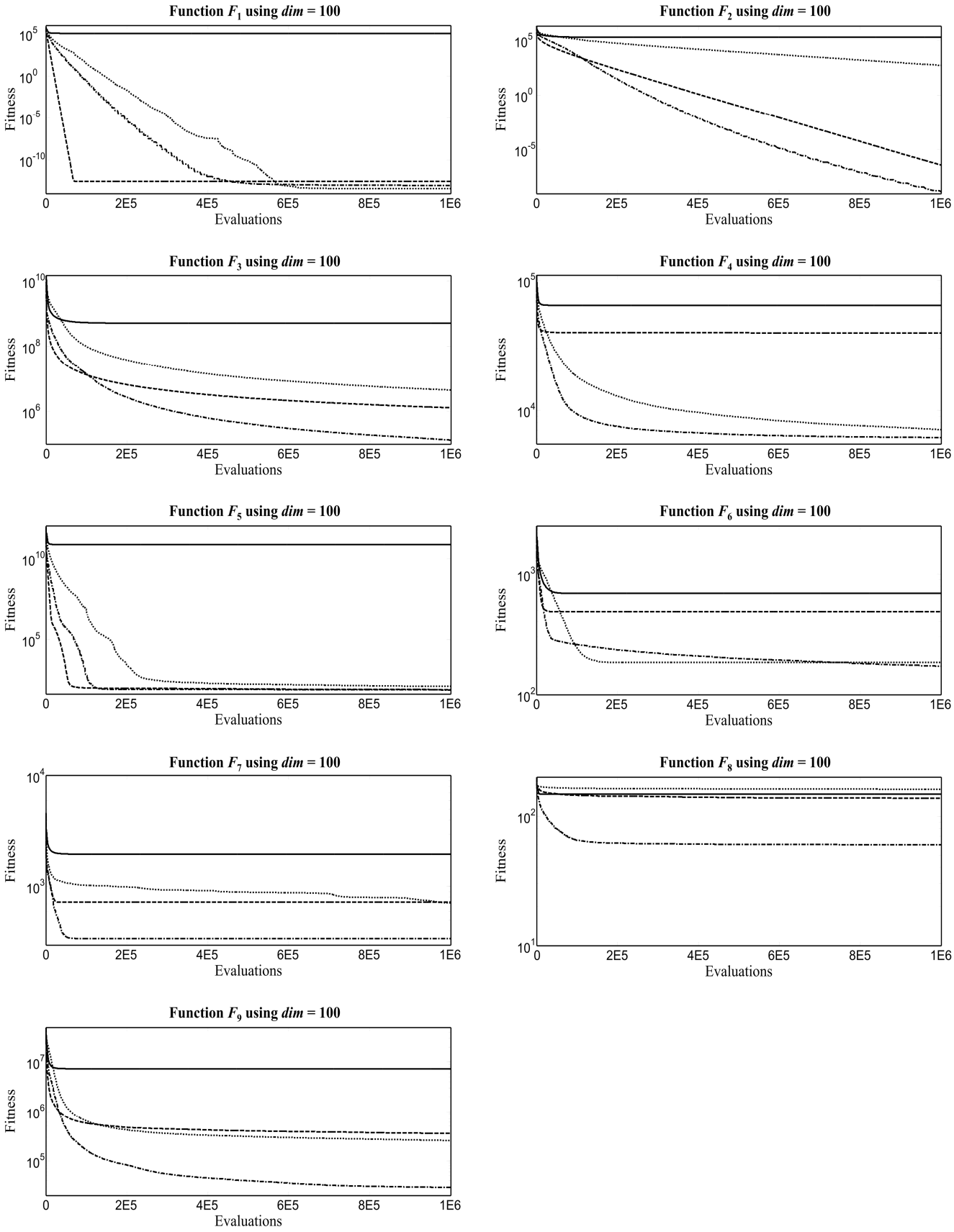


Fig. 6. *Evolution of the average value of the best solution found by ICA-NM, ICA, SPSO2011 and DE for the CEC 2005 benchmark functions using* $dim = 100$. *A solid line is used for ICA, a dashed line is used for SPSO2011, a dotted line is used for DE and a dash-dot line is used for ICA-NM. For more clarity, we used a logarithmic scale for the ordinates.*

To conclude, the optimization problem consists in optimizing two given objective functions:

Table 2. *The main parameters used in the optimization process of the OR-PMSM.*

Symbol	Description	Value	Variation limits
Dir	Rotor inner diameter	207 mm	[185; 210] mm
hjr	Rotor yoke height	8 mm	[5; 10] mm
his	Tooth isthmus height	2 mm	[1; 3] mm
hjs	Stator yoke height	6.5 mm	[5; 10] mm
ld	Tooth width	7 mm	[4; 10] mm
gap	Air-gap length	1 mm	[0.5; 1.5] mm
hmp	PM height	3 mm	[2.5; 7] mm
Lm	Machine's length	50 mm	[30; 80] mm
Tm	Motor torque	34.1 Nm	[34; 34.2] Nm
Pout	Output power	1500 W	[1490; 1510] W
Is	Motor phase current	21 A	[15; 21.5] A

1. The first objective function concerns the minimization of the mass of the active parts of the machine, here called m_{tot}. The total mass of the active parts of the machine needs to be minimized. m_{tot} represents the sum of all active parts of the machine:

$$m_{tot} = m_{copper} + m_{stat} + m_{rot} + m_{PM} \quad (13)$$

where m_{copper} is the mass of copper used for windings, m_{stat} is the mass of the stator core (teeth and yoke), m_{rot} is the mass of the rotor core and m_{PM} is the mass of the PMs.

2. The second objective function consists in the maximization of the output power P_{out} which is equal to the difference between the input power and the losses:

$$P_{out} = P_{in} - \sum Losses \quad (14)$$

The sum of losses contains mainly the iron and copper losses; the mechanical losses (usually estimated at 0.5% of the output power) are neglected here. The objective function to optimize using our algorithm is represented by:

$$Minimize f_{motor} = -RPM + penality \quad (15)$$

with

$$RPM = \frac{P_{out}}{m_{tot}}$$

$$penality = \begin{cases} 0 & \text{if all constraints are respected} \\ 10^7 \times \sum_{i=1}^{3} \frac{|C_i - Limit_i|}{|\text{Upper Bound}(i) - \text{Lower Bound}(i)|} & \text{otherwise} \end{cases}$$

and

$$Limit_i = \begin{cases} \text{Lower Bound}(i) & \text{if } C_i < \text{Lower Bound}(i) \\ \text{Upper Bound}(i) & \text{if } C_i > \text{Upper Bound}(i) \end{cases}$$

4.3. Parameter Setting

The values of the ICA and ICA-NM parameters used for the CEC 2005 benchmark, as well as for the electric machine design problem, are defined in Table 3. They have been determined empirically for these problems.

Table 3. *ICA and ICA-NM parameters.*

	Parameter	Value	
		CEC 2005	Motor design
ICA	Number of initial countries N_{pop}	24	100
	Number of Initial Imperialists N_{imp}	8	6
	Assimilation coefficient β	2	2
	Assimilation angle coefficient γ	0.4	0.35
	Coefficient ξ used in (4)	0.05	0.065
	UnitingThreshold used to merge empires	0.02	0.005
ICA-NM only	δ_t used in the starting criterion of NM	15	50
	c_{it} (to initialize the first run of NM)	10	60
	c_{damp} (to initialize the next runs of NM)	0.96	0.99

The performances of ICA-NM are not only compared to those of ICA, but also to those of other well-known optimization algorithms. These algorithms and their parameter setting, empirically fitted, are defined below (see references for more details on these algorithms and their parameter fitting).

- SPSO2011 (*Standard Particle Swarm Optimization* in its 2011 version) [10] using a number of particles equal to $S = 24$, an inertia weight equal to $w = 0.721$, learning factors c_1 and c_2 equal to 1.193, and $K = 3$ for the parameter K used to generate the particles neighborhood ;

- DE (*Differential Evolution*) [11] using the *DE/rand/1/bin* strategy, a population size equal to $NP = 24$, a weighting factor $F = 0.8$, and a crossover constant $CR = 0.9$.

5. Experimental Results and Discussion

5.1. Results for the CEC 2005 Benchmark

The convergence of ICA and ICA-NM are studied and compared in Fig. 5 and Fig. 6 for the CEC 2005 benchmark functions using $dim = 50$ and $dim = 100$. For each function, the evolution of the fitness of the best solution found by an algorithm is presented, averaged over 20 runs. As one can see on each graph, the solution found by ICA-NM is better than the one found by ICA for almost all evaluations.

Moreover, ICA, ICA-NM, SPSO2011 and DE are run 20 times using each benchmark function. The average value of the best solution found by each algorithm at the end of its execution, with its standard deviation, is presented in Table 4 for every function. The Kruskal-Wallis statistical test has been used to determine if a significant difference exists between the results obtained by the algorithms. This test indicates at 95% confidence level that there is a significant difference between the performances of the algorithms for all test cases. Then, the Tukey-Kramer post hoc test is used to determine which algorithms perform differently from ICA-NM. The results of this test are presented in Table 4, where the letter *W* appears under the results that are significantly *worse* than the ones of ICA-NM, and the letter *B* appears under the results that are significantly *better* than the ones of ICA-NM. As one can see, hybridizing ICA with the NM simplex method significantly improves the performance of the algorithm for all test cases. Compared to SPSO2011, ICA-NM is able to produce better results for 16 test cases, and similar ones for the 2 others. Compared to DE, ICA-NM is able to produce better results for 9 test cases, and similar ones for 7 test cases. It is only outperformed by DE for 2 test cases among 18 (F_1 using $dim = 50$ and $dim = 100$).

5.2. Results for the Electric Motor Design Problem

The results obtained by ICA-NM for the motor design problem are compared to those of ICA and several well-known multiobjective algorithms: ASREA [12], AMGA [13] and OMNIOPT [14]. The parameters of each algorithm have been fitted to the problem experimentally. Each algorithm is run 100 times, and the best solution found during these runs is kept. The maximum number of evaluations is set to 1E+5 for each algorithm. These solutions are presented in Table 5, along with the reference solution for this problem, given in [9]. As we can see, every algorithm is able to produce a better solution than the reference one. However, the solution found by ICA-NM is better than the ones found by the other algorithms. The use of ICA-NM enables a mass reduction of the motor of 46.5% with regard to the reference solution.

To further compare the performance of ICA and ICA-NM for this problem, the values of *RPM* found by ICA and ICA-NM, averaged over 100 runs of each algorithm, are computed. All the solutions found by both algorithms are feasible. The average value and standard deviation of *RPM* obtained by ICA and ICA-NM are 346.22 ± 19.45 and 378.80 ± 18.20, respectively. The Friedman statistical test indicates at 95% confidence level that there is a significant difference between the results of ICA and ICA-NM for this problem. As one can see, the performance of ICA-NM is significantly better than the one of ICA for this problem.

The convergence of both algorithms is compared in Fig. 7, where the evolution of the average value of $f_{motor} + 400$ over 100 runs of each algorithm is illustrated. We add 400 to the value of the objective function in order to enable the use of a logarithmic scale for the ordinates. As one can see, the solution found by ICA-NM is better than the one found by ICA for almost all evaluations. The Friedman statistical test indicates at 95% confidence level that there is a significant difference between the results of ICA and ICA-NM from the 5580^{th} evaluation to the last one. Hence, ICA-NM is able to produce significantly better solutions than ICA from the 5580^{th} evaluation.

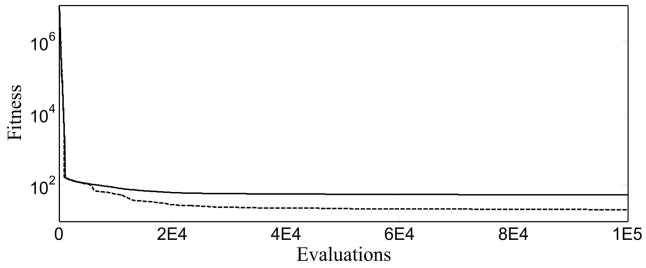


Fig. 7. *Evolution of the average value of the best solution found by ICA and ICA-NM for the electric motor design problem. A solid line is used for ICA, and a dashed line is used for ICA-NM. For more clarity, the values are shifted by 400 in order to use a logarithmic scale for the ordinates.*

5.3. Validation of the Optimization Algorithm Via Experimental Results Measured on the Constructed Prototype

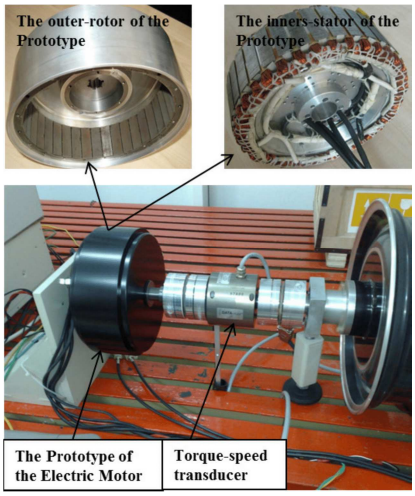
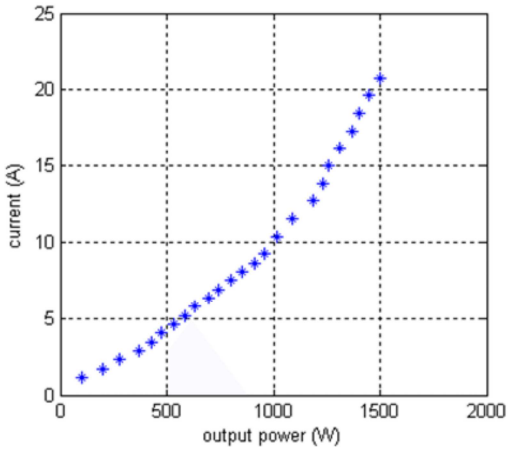


Fig. 8. *The experimental bench for testing the constructed prototype.*

We have constructed a prototype of the electric motor and we have tested on a measurement bench.

The test bench of the studied in-wheel motor used for the motorization of an electric scooter is presented in Fig. 8. Here, one can distinguish the constructed prototype. The main parts of the prototype (which were optimized), meaning the rotor and the stator, are presented also in Fig. 8-top. The mechanical performances are measured with a torque-speed transducer (DataFlex22). The load is assured via an electric machine and the control and the data acquisition are used with a dSPACE1103, board installed on a PC. The converter is made of IRFZ48-mosfet switches.



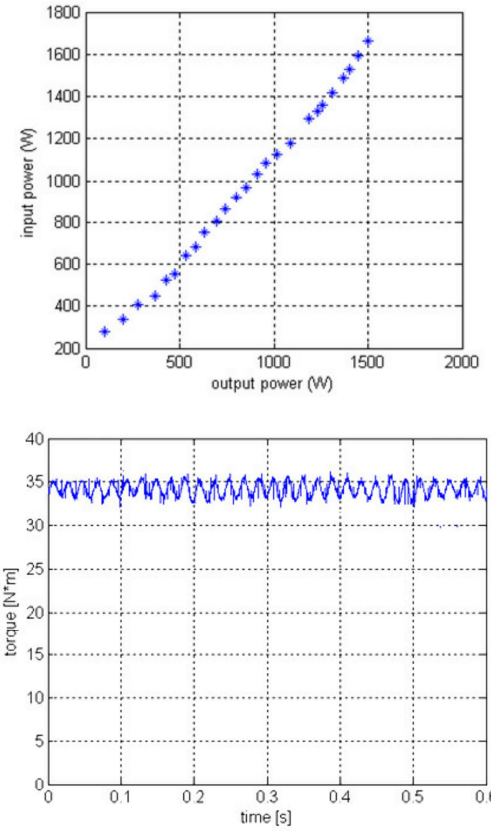


Fig. 9. *The measured results of the electric motor.*

Table 4. *Average value and standard deviation of the best solution found by ICA-NM, ICA, SPSO2011 and DE for each benchmark function. The letter W is written under the results that are significantly worse than the ones of ICA-NM. The letter B is written under the results that are significantly better than the ones of ICA-NM. No letter appears under the results that are not significantly different.*

dim	Function	Fitness ± Standard deviation			
		ICA-NM	**ICA**	**SPSO2011**	**DE**
50	F_1	4.72E−14 ± 1.04E−14	2.01E+04 ± 5.89E+03 *W*	1.26E−13 ± 3.79E−14 *W*	2.05E−14 ± 9.90E−15 *B*
	F_2	3.31E−13 ± 9.21E−14	5.06E+04 ± 2.70E+04 *W*	1.81E−12 ± 4.16E−13 *W*	4.52E−02 ± 3.47E−02 *W*
	F_3	1.59E+04 ± 4.93E+03	1.16E+08 ± 9.39E+07 *W*	2.88E+05 ± 8.50E+04 *W*	1.01E+06 ± 3.01E+05 *W*
	F_4	3.31E+03 ± 3.23E+02	2.70E+04 ± 4.77E+03 *W*	1.53E+04 ± 2.26E+03 *W*	2.78E+03 ± 4.63E+02
	F_5	1.90E+02 ± 2.86E+02	9.98E+09 ± 6.28E+09 *W*	2.56E+02 ± 2.34E+02	4.22E+01 ± 3.91E+01
	F_6	4.23E+01 ± 1.31E+01	2.55E+02 ± 6.10E+01 *W*	1.62E+02 ± 2.24E+01 *W*	6.94E+01 ± 1.88E+01
	F_7	1.30E+02 ± 2.68E+01	6.29E+02 ± 1.12E+02 *W*	2.17E+02 ± 5.23E+01 *W*	3.00E+02 ± 1.16E+02 *W*
	F_8	2.63E+01 ± 6.80E+00	6.49E+01 ± 3.57E+00 *W*	5.56E+01 ± 6.79E+00 *W*	7.12E+01 ± 1.01E+01 *W*
	F_9	1.17E+04 ± 9.76E+03	1.31E+06 ± 3.30E+05 *W*	4.71E+04 ± 4.04E+04 *W*	3.54E+04 ± 3.32E+04
100	F_1	9.38E−14 ± 8.03E−15	1.15E+05 ± 2.78E+04 *W*	2.94E−13 ± 4.90E−14 *W*	3.74E−14 ± 2.35E−15 *B*
	F_2	1.52E−09 ± 1.32E−09	1.93E+05 ± 6.57E+04 *W*	3.97E−07 ± 2.13E−07 *W*	5.29E+02 ± 2.02E+02 *W*
	F_3	1.33E+05 ± 3.50E+04	5.19E+08 ± 2.50E+08 *W*	1.29E+06 ± 3.66E+05 *W*	4.48E+06 ± 1.52E+06 *W*

dim	Function	Fitness ± Standard deviation			
		ICA-NM	ICA	SPSO2011	DE
	F_4	6.16E + 03 ± 1.01E + 03	6.44E + 04 ± 7.79E + 03 *W*	3.96E + 04 ± 3.86E + 03 *W*	7.10E + 03 ± 1.50E + 03
	F_5	8.91E + 01 ± 1.91E + 00	7.26E + 10 ± 3.40E + 10 *W*	9.32E + 01 ± 1.99E + 01	1.49E + 02 ± 5.69E + 01
	F_6	1.72E + 02 ± 2.14E + 01	6.98E + 02 ± 1.22E + 02 *W*	4.92E + 02 ± 7.33E + 01 *W*	1.84E + 02 ± 4.44E + 01
	F_7	3.43E + 02 ± 4.05E + 01	1.98E + 03 ± 2.22E + 02 *W*	7.24E + 02 ± 9.15E + 01 *W*	7.13E + 02 ± 3.30E + 02 *W*
	F_8	6.03E + 01 ± 1.20E + 01	1.48E + 02 ± 6.67E + 00 *W*	1.37E + 02 ± 7.95E + 00 *W*	1.61E + 02 ± 1.51E + 00 *W*
	F_9	2.91E + 04 ± 2.44E + 04	7.32E + 06 ± 1.77E + 06 *W*	3.56E + 05 ± 2.20E + 05 *W*	2.56E + 05 ± 1.49E + 05 *W*

Table 5. *Properties of the best solution found by each algorithm among 100 runs.*

Property	Reference solution	ICA	ICA-NM	ASREA	AMGA	OMNIOPT
m_{tot} (kg)	7.1207	3.9097 − 45.094 (%)	3.8097 − 46.498 (%)	5.8946 − 17.218 (%)	4.2863 − 39.805 (%)	3.8652 − 45.718 (%)
RPM (W/kg)	210.67	385.18 + 82.84 (%)	391.37 + 85.77 (%)	255.09 + 21.08 (%)	347.90 + 65.14 (%)	389.47 + 84.87 (%)

To validate the design and the optimization algorithm, and to check if the prototype satisfies the application's demands, we will check mainly the output power, the current level and the torque of the machine (based on the input and output power, one can get also the efficiency of the motor).

The measured results are presented in Fig. 9, being obtained by load variation, from zero up to the rated torque.

The load current at rated operation is 21.05 A, which is very close to the analytically obtained value. Based on the input (1665 W) and output (1500 W) power, we can get the efficiency of our motor: 90.1%, also very close to the analytical value. Also, we can see that the desired torque (34 Nm) is obtained.

Thus, we can conclude that the designed and optimized electric motor satisfies the application demands.

6. Conclusion

A new hybrid algorithm that combines an Imperialist Competitive Algorithm (ICA) with the Nelder-Mead simplex method (NM) has been proposed. This algorithm, called ICA-NM, has been designed in order to improve both diversification and intensification capabilities of ICA. A comparison of the performances of ICA-NM with the ones of ICA and several well-known metaheuristics has been presented, using benchmark functions of the CEC 2005 congress. This comparison shows the efficiency of our hybridization. Then, ICA-NM is used to design a motor for an electric scooter. It is able to produce better results than several multiobjective algorithms used for this practical problem. The solution found by ICA-NM leads to a 46.5% mass reduction of the motor with regard to the reference solution.

In works in progress, we are working on the integration of machine learning techniques to ICA-NM in order to guide the search towards possible more promising areas of the search space. The critical parameters of ICA-NM could also be automatically adjusted through this kind of techniques. Besides, we also plan to apply ICA-NM to other real-world problems.

References

[1] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *Proceedings of IEEE Congress on Evolutionary Computation*, Singapore, September 2007, pp. 4661–4667.

[2] J. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[3] E. Zahara, S.-K. S. Fan, and D.-M. Tsai, "Optimal multi-thresholding using a hybrid optimization approach," *Pattern Recognition Letters*, vol. 26, no. 8, pp. 1082–1095, 2005.

[4] J. Dréo and P. Siarry, "An ant colony algorithm aimed at dynamic continuous optimization," *Applied Mathematics and Computation*, vol. 181, no. 1, pp. 457–467, 2006.

[5] A. Liu and M.-T. Yang, "A new hybrid nelder-mead particle swarm optimization for coordination optimization of directional overcurrent relays," *Mathematical Problems in Engineering*, vol. 2012, pp. 1–18, 2012.

[6] M. Joorabian and E. Afzalan, "Optimal power flow under both normal and contingent operation conditions using the hybrid fuzzy particle swarm optimisation and Nelder-Mead algorithm (HFPSO-NM)," *Applied Soft Computing*, vol. 14, no. 0, pp. 623–633, 2014.

[7] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological University, ETH Zurich, Indian Institute of Technology, National Chiao Tung University, Tech. Rep. 2005005, 2005.

[8] W. Spendley, G. R. Hext, and F. R. Himsworth, "Sequential application of simplex designs in optimization and evolutionary operation," *Technometrics*, vol. 4, pp. 441–461, 1962.

[9] S. Giurgea, D. Fodorean, G. Cirrincione, A. Miraoui, and M. Cirrincione, "Multimodel optimization based on the response surface of the reduced FEM simulation model with application to a PMSM," *IEEE Transactions on Magnetics*, vol. 44, no. 9, pp. 2153–2157, 2008.

[10] M. Clerc et al., "The *Particle Swarm Central* website," http://www.particleswarm.info, 2014.

[11] K. Price, R. Storn, and J. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*. Springer, 2005.

[12] D. Sharma and P. Collet, "An archived-based stochastic ranking evolutionary algorithm for multiobjective optimization," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. Portland, Oregon, USA: ACM, July 2010, pp. 479–486.

[13] S. Tiwari, G. Fadel, P. Koch, and K. Deb, "Performance assessment of the hybrid archive-based micro genetic algorithm on the CEC09 test problems," in *Proceedings of IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 2009, pp. 1935–1942.

[14] K. Deb and S. Tiwari, "Omni-Optimizer: A generic evolutionary algorithm for single and multiobjective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008.