Parameter Identification of Induction Motors using Stochastic Optimization Algorithms

Rasmus K. Ursem [a], Pierré Vadstrup [b]

[a] *EVALife, Department of Computer Science, University of Aarhus, Bldg. 540, Ny Munkegade, DK-8000 Aarhus C, Denmark*

[b] *GRUNDFOS Management A/S, Poul Due Jensens Vej 7, DK-8850 Bjerringbro, Denmark*

Abstract

In the area of optimization, applied research typically focuses on finding the best possible solution to the practical problem at hand. In contrast, a large part of basic research aims at developing novel algorithms with improved performance. In practical application, most studies employ rather simple algorithms. On the other hand, most novel algorithms are tested on a suite of artificial benchmark problems. Hence, there is a research gap between applied and basic research since the former focuses on the *problems*, whereas the latter concentrates on the *algorithms*.

In this paper, we address the research gap for parameter identification of induction motors. In our experiments, we compare the performance of eight stochastic optimization algorithms on identification of two induction motors. The eight algorithms represent four main groups of algorithms presently used for numerical optimization. The four groups are: local search (LS), evolution strategies (ESs), generational evolutionary algorithms (EAs), and particle swarm optimizers (PSOs). The comparison includes a simple and an advanced algorithm from each group.

From our experiments, we draw two conclusions. First, the advanced algorithms had significantly better performance compared with the simple algorithms. This underlines the importance of using advanced algorithms when approaching real-world problems. Furthermore, the improved performance justifies and motivates the development of more advanced techniques. Second, population-based stochastic optimization techniques (ESs, EAs, and PSOs) significantly outperformed the local search algorithms on both problems.

Key words: induction motor, parameter identification, local search, evolution strategies, evolutionary algorithms, particle swarm optimizers

Email addresses: ursem@daimi.au.dk (Rasmus K. Ursem), pvadstrup@grundfos.com (Pierré Vadstrup).

1 Introduction

A fundamental part of control engineering is the identification of the system being controlled. In system identification, considerable efforts have been devoted to develop methods for identification of system models and their parameters. Currently, a wide range of analytical techniques exists for linear systems. An overview of the methods used in the control area for system identification can be found in Ljung's book [15]. For non-linear systems, limited progress has been made with analytical approaches. Instead, some success has been achieved with various traditional optimization methods such as least squares and local search. However, a fundamental problem of traditional optimization techniques is their dependence on unrealistic assumptions such as unimodal performance landscapes and differentiability of the performance function [1] . Consequently, non-linear problems are often oversimplified to fulfill such assumptions, which makes the found solutions hard to transfer back to the original problem. Evolutionary algorithms (EAs) and other stochastic search techniques seem to be a promising alternative to traditional techniques. First, EAs do not rely on any assumptions such as differentiability, continuity, or unimodality. Second, they are capable of handling problems with non-linear constraints, multiple objectives, and time-varying components. Third, they have shown superior performance in numerous real-world applications.

In this paper, we investigate two induction motors (1.1 kW and 5.5 kW) used in the pumps produced by Grundfos A/S. The main objective in this paper is to evaluate a number of stochastic search algorithms with respect to parameter identification of two induction motors. By this approach, we aim at addressing the gap between applied and basic research [2] . In most practical applications, the focus is entirely on the problem, and typically a simple algorithm is used to find a reasonable solution. This is in clear contrast to the numerous investigations on artificial numerical optimization problems showing that almost any extension of the basic algorithms leads to a significant performance improvement. However, an important criterion for choosing a more advanced algorithm is the extra effort necessary to implement it, which should be weighted against the potential performance improvement. The advanced algorithms used in our comparison have been selected with this in mind, i.e., the extensions in these algorithms are small but have lead to significant improvement in performance for artificial and real-world problems in other practical areas of research.

The paper is organized as follows. Section 2 provides a survey of EA-investigations in parameter identification of induction motors. The eight algorithms

[1] In the EA-community, the performance function is called the fitness function.
[2] To the authors experience, this gap exists in many areas where stochastic optimization algorithms are applied to real-world problems.

are described in Section 3. In Section 4, we introduce the two induction motor models and define the performance function (fitness). The experimental setup and the results are covered in Section 5. Finally, the outcome of the experiments is discussed in Section 6.

2 Literature review

Induction motors have been intensively investigated and described in the engineering literature for several decades. In this context, a number of non-linear models of the induction motor that incorporate magnetic saturation effects have been suggested; for instance by Vas [22]. In spite of the considerable theoretical foundation of induction motors, few studies have used EAs and other stochastic search techniques to identify the model parameters. However, some investigations have been presented relying on models not accounting for non-linear effects. These methods are primarily used for control purposes and they recursively estimate the parameters by use of various methods – mainly Extended Kalman Filters. See [4], [18], and [24] for further information on these methods. EA-investigations on parameter identification of induction motors can be categorized in two groups.

(1) Identification of parameters in simulation models based on non-linear differential equations.
(2) Identification of parameters from equations describing the load torques.

The approach used in the first group determines the parameters from time-series data, which typically leads to high-accuracy models that can be used to control the motor. The approach used in the second group is less precise, but is independent of available time-series data. This approach is often used in more general system studies. The study presented in this paper belongs to the first group. In this context, only a few studies have been published that use EAs to determine the parameters of high-accuracy simulation models. In a recent study, Alonge et al. investigated a 1 kW motor and showed that the evolutionary algorithm GENESIS outperformed least squares fitting [3]. Alonge et al. determined stator resistance (R_s), stator inductance (L_s), leakage inductance (L_e), motor load (τ_r), and moment of inertia (J_m) using a state space model with scaled rotor flux. Their study is a followup of an earlier published investigation [2]. In a similar study, Huang et al. identified stator and rotor resistance (R_s and R_r), stator and rotor self inductance (L_s and L_r), and magnetizing inductance (L_m) using a motor model not accounting for saturation effects [12]. In their paper, Huang et al. compared the performance of a genetic algorithm (GA) to a random search algorithm under four levels of simulated measurement noise. In a more distantly related study, Ju and Handschin determined the parameters in a motor load model using a GA [9].

Regarding investigations belonging to the second group, Haque et al. used a simple evolutionary algorithm to determine stator resistance (R_s), rotor resistance (R_r), and combination of stator and rotor reactance (X_{lr}) from the motor's specifications provided by the manufacturer [10]. In a followup study, Nangsue et al. used a GA to determine the parameters of three motors from equations for the full load torque, the lock rotor torque, and the breakdown torque [17]. Additionally, Nangsue et al. used genetic programming (GP) to evolve equations modeling these torques. Their results are somewhat inconclusive since the GA outperformed the GP in some cases, but not in other cases.

As mentioned above, this study belongs to the first group of high-accuracy model identification. The two other studies by Alonge et al. [2,3] and Huang et al. [12] both focus on finding the best parameters, but not on determining the best *algorithm* to achieve this task, which is typically the objective in more basic research on evolutionary computation (EC). Hence, the mentioned research gap between practical application and basic EC-research exist in the case of parameter identification of induction motors. In this investigation, we aim at bridging this gap in this field of research.

3 Algorithms

In our experiments, we compare eight algorithms taken from four main groups of stochastic search algorithms. The comparison includes a simple algorithm and an advanced algorithm from each group. Selecting the algorithms to include in a comparison is difficult, especially when considering the rather large number of new algorithms suggested every year. The algorithms compared in this study have been selected by considering their potential performance improvement versus their complexity regarding the implementation of the algorithm. In this selection, the features (multimodality, parameter correlation, etc.) of the investigated problems have not been considered, because these are typically unknown prior to the experimentation and only partially known after. To ensure a broad comparison, we have chosen algorithms from the following four groups:

(1) Local search algorithms.
(2) Evolution strategies.
(3) Generational evolutionary algorithms.
(4) Particle swarm optimizers.

These groups represent four main types of stochastic algorithms currently used for numerical optimization problems. The following sections describe the algorithms in detail.

3.1 Local Search Algorithms

The basic idea in local search algorithms is to iteratively improve a single solution by looking in its neighborhood and choose the most promising adjacent solution as a new candidate. A local search algorithm usually starts with a randomly generated solution and iterates until no improvement occurs or until some termination criterion is met, e.g., the maximal number of evaluations is reached or the solution is sufficiently good. The advantage of local search algorithms is their simplicity. The drawback is their inappropriateness for multimodal problems, because they have difficulties escaping local optima (deterministic local search cannot escape).

The steepest descent local search algorithm (SDLS) is probably one of the most frequently used local search algorithms. To apply this algorithm, the continuous search space of n-dimensional numerical problems needs to be discretized. This is done by setting a step-size δ_i for each problem variable x_i. The neighborhood $\mathcal{N}_{\mathbf{x}}$ of a solution $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is then usually defined as the set of solutions reachable from $\mathbf{x}$ by adding or subtracting *one* δ_i from $\mathbf{x}$, i.e., $\mathcal{N}_{\mathbf{x}} = \{(x_1 \pm \delta_1, x_2, \ldots, x_n), (x_1, x_2 \pm \delta_2, \ldots, x_n), \ldots, (x_1, x_2, \ldots, x_n \pm \delta_n)\}$. The neighborhood size is equal to $2n$ using this definition.

Simulated annealing (SA) is a variant of local search where the convergence criterion is relaxed to make the algorithm capable of escaping local optima. In SA, inferior solutions are accepted with a probability depending on the algorithm's parameter called temperature and the difference between the current solution and the new candidate solution. The probability of accepting inferior solutions decrease as the algorithm progresses. In the later presented experiments, we used the algorithm defined by Michalewicz and Fogel [16, p. 120] with linearly decreasing temperature from T_{max} to T_{min}. For more information on SA and other local search techniques, see [14], [16] and [8].

The parameters for the algorithms are listed in Table 1.

Algorithm	Parameter	Value	Description
SDLS	No parameters except step-sizes (see Section 5)		
SA	MAX-TRIES	10	Max tries between change of temperature
	T_{max}	20	Start temperature
	T_{min}	10	End temperature

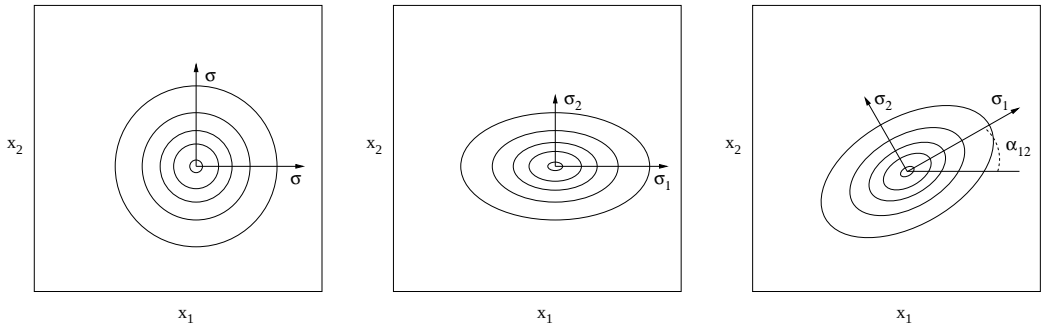
Table 1
Parameters for steepest descent local search and simulated annealing.

3.2 Evolution Strategies

Evolution strategies (ES) were introduced by Rechenberg and Schwefel in 1964. The first version of the algorithm was a so-called (1+1)-ES, in which one parent created one offspring by Gaussian mutation [3] [19]. The offspring replaced the parent if it had a better fitness. In later studies, ES have been extended with a population of μ parents creating λ offspring using both mutation and recombination. The main difference between ES and the generational EAs is the selection procedure. Evolution strategies use deterministic selection whereas selection in generational EAs is probabilistic. Currently, two main selection strategies exist in ES; the $(\mu + \lambda)$-ES and the (μ, λ)-ES. In $(\mu + \lambda)$-ES, the μ parents create λ offspring. The next population is then formed by deterministically selecting the μ best individuals among the available $\mu + \lambda$ individuals. The number of offspring λ is usually less than the total population size, which gives an algorithm with overlapping populations. A typical setting is $\mu = 100$ and $\lambda = 15$, which will replace a maximum of 15 individuals in each iteration. The other strategy, (μ, λ)-ES, also generates λ individuals from the μ parents. However, in (μ, λ)-ES the parent population is not included in the source population in the selection procedure. The population in (μ, λ)-ES is therefore non-overlapping. Hence, λ must be larger than μ, because individuals are not cloned in ES [4]. The Gaussian mutation operator is the main component for creating new solutions in ES. In this context, most ES algorithms use self-adaptation to adjust the search process to the problem. The idea in self-adaptation is to encode algorithmic parameters in the genome and use these parameters to modify the individual. The hypothesis is that good solutions carry good parameters; hence, evolution discovers good parameters *while* solving the problem. Simple self-adaptation only encodes one variance σ for all problem variables (see Figure 1(a)). A more advanced self-adaptation scheme encodes one variance σ_i for each variable (Figure 1(b)). Another variant supports correlation between problem variables by encoding both variances σ_i and an additional set of rotation angles α_{ij} (Figure 1(c)). For additional information on the self-adaptive mutation operator, see [6, Section 6.4] and [5].

Our comparison includes the simple self-adaptive $(\mu+\lambda)$-ES with one encoded variance σ and the complex version using self-adaptation of both variances σ_i and rotation angles α_{ij}. The latter was chosen since it is capable of automatically discovering parameter correlations. For comparison, all four evolutionary algorithms (the two evolution strategies and the two generational EAs) used the same crossover operator. The crossover operator had one weight for each

[3] In Gaussian mutation, each problem parameter is altered by adding a normal distributed number, which is generated anew for each parameters.
[4] A ratio of $\lambda/\mu \approx 7$ is recommended in [6, Section 6.4].



(a) Self-adaptation of one variance σ.

(b) Self-adaptation of all variances σ_i.

(c) Self-adaptation of variances σ_i and rotation angles α_{ij}.

Figure 1. Variants of self-adaptive mutation operator for evolution strategies. The boxes denote the search space. Ellipses are level-curves with equal mutation probability density.

variable, and all weights except one were randomly assigned to either 0 or 1. The single remaining weight was set to a random value from the interval $[0, 1]$. Hence, the crossover operator is a mix between the traditional uniform and arithmetic crossover operators. This approach was chosen since it showed significantly better performance in preliminary experiments.

The parameters for the two variants of ES are listed in Table 2.

Algorithm	Parameter	Value	Description
ES1, ES2	μ	100	Parent population size
	λ	15	Offspring population size
	p_c	0.5	Crossover probability
	σ_0	1.0	Initial σ value
	τ_0	2.0	Variance for $z_0 \sim N(0, \tau_0^2)$ vector [6, Eq. 6.18]
	τ	2.0	Variance for $z_i \sim N(0, \tau^2)$ vector [6, Eq. 6.18]
	β	0.0873	Variance for $z_i \sim N(0, \beta^2)$ vector [6, Eq. 6.19]
	ε_σ	1.0E-5	Min σ value

Table 2
Parameters for the two variants of the evolution strategy. The algorithms used the same parameter values.

3.3 Generational Evolutionary Algorithms

The generational evolutionary algorithms (EAs) differ from evolution strategies on three aspects. First, the populations in the iterations are considered as a number of consecutive generations, i.e., the population at time 1 breeds

the population at time 2. Hence, the populations are non-overlapping. Second, selection is stochastic rather than deterministic. Third, individuals are cloned in the selection process. Consequently, multiple copies of the better fit individuals are typically present in the population after selection. One of the first generational EAs was Hollands genetic algorithm [11], which used binary strings to encode the individuals.

In this study, we included a simple evolutionary algorithm (SEA) [5] in the comparison. The SEA encoded solutions as real-valued vectors and used Gaussian mutation with zero mean and annealing variance $\sigma^2 = 1/\sqrt{\mathtt{t}+1}$ ($\mathtt{t}$ is the generation number). The mutation operator scaled the randomly generated numbers by 10% of the length of the search intervals to make the operator independent of the variables' ranges. Thus, a value generated according to $N(0,\ 0.1 \cdot l_j \cdot \sigma^2)$ was added to the j'th parameter x_{ij} of the i'th solution x_i (l_j is the length of the j'th search interval). The mutation on all parameters is performed with probability p_m, which was 0.75 in the experiments (see Table 3). Recombination was performed as described in the previous section. The next generation was selected using binary tournament selection, which, for each slot in the new population, copies the best of two randomly selected individuals. An elitist strategy keeping the best individual was enforced to ensure that the best known solution survived to the next generation.

In addition to the SEA, we included the recently introduced diversity-guided evolutionary algorithm (DGEA) [21]. The idea behind the DGEA is simple. Unlike most other EAs, the DGEA uses a diversity measure to alternate between exploring and exploiting behavior. To use a measure for this purpose, it has to be robust with respect to the population size, the dimensionality of the problem, and the search interval length of each of the variables. An immediate measure for n-dimensional numerical problems is the "distance-to-average-point" measure defined as:

$$diversity(P) = \frac{1}{|L| \cdot |P|} \cdot \sum_{i=1}^{|P|} \sqrt{\sum_{j=1}^{n} (x_{ij} - \overline{x}_j)^2} \tag{1}$$

where $|L|$ is the length of the diagonal [6] in the search space $\mathcal{S} \subseteq \mathbb{R}^n$, P is the population, $|P|$ is the population size, n is the dimensionality of the problem, x_{ij} is the j'th value of the i'th individual, and $\overline{x}_j$ is the j'th value of the average point $\overline{\mathbf{x}} = \frac{1}{|P|} \sum_{i=1}^{|P|} x_i$, where x_i is the i'th individual.

The DGEA applies *diversity-decreasing* operators (selection and recombina-

[5] The term "standard evolutionary algorithm" is sometimes used in EA-literature. However, to the authors belief there is no such thing as a standard EA, because of the wide variety in available encodings and operators.

[6] Assuming that each search variable x_k is in a finite range, $x_{kmin} \leq x_k \leq x_{kmax}$.

tion) as long as the diversity is above a certain threshold d_{low}. When the diversity drops below d_{low} the DGEA switches to *diversity-increasing* operators (mutation) until a diversity of d_{high} is reached. Hence, phases with exploitation and phases with exploration will occur. Theoretically, the DGEA should be able to escape local optima, because the algorithm will force higher diversity regardless of fitness. The DGEA was introduced with constant values for d_{low} and d_{high} [21]. However, preliminary experimentation performed in this study showed that a simple linearly decreasing value significantly improved the performance of the algorithm. To this end, d_{lowmax} and d_{lowmin} was used to calculate d_{low} (likewise for d_{high}). Furthermore, forcing a switch to exploration if the fitness stagnated proved to be beneficial. This forced exploration is controlled by $d_{staglow}$ and $d_{stagmax}$. Generations with no fitness improvement are counted when the diversity is below $d_{staglow}$. An exploration phase is enforced if this counter reaches $d_{stagmax}$. A fitness improvement resets the stagnation counter.

The parameters for the SEA and the DGEA are displayed in Table 3.

Algorithm	Parameter	Value	Description
SEA	ps	100	Population size
	p_m	0.75	Mutation probability
	p_c	0.9	Crossover probability
DGEA	ps	100	Population size
	p_m	0.75	Mutation probability
	p_c	0.9	Crossover probability
	λ	1.0	Mutation parameter
	d_{scale}	0.02	Scale factor
	d_{lowmax}	1.0E-5	Start value for d_{low}
	d_{lowmin}	5.0E-12	End value for d_{low}
	$d_{highmax}$	0.1	Start value for d_{high}
	$d_{highmin}$	0.01	End value for d_{high}
	$d_{staglow}$	0.0005	Diversity for stagnation counting
	$d_{stagmax}$	20	Max no. of stagnated generations

Table 3
Parameters for the simple evolutionary algorithm and the diversity-guided evolutionary algorithm.

3.4 Particle Swarm Optimizers

Particle swarm optimizers (PSOs) are partly inspired by the behavior of large animal swarms such as schooling fish or flocking birds. The main idea is to interpret each particle as a solution to the problem being investigated and to

let these particles explore the search space. A good analogy is to imagine a swarm of mosquitoes being attracted to a light bulb. The particles are the mosquitoes and the bulb is the current best optimum, which changes as better optima are discovered during the optimization procedure. Particle swarm optimization is a rather new technique introduced by Kennedy and Eberhart in 1995 [13].

In the standard particle swarm optimizer (STDPSO), each particle i has a position ($\mathbf{x}_i$) in the search space, a velocity vector ($\mathbf{v}_i$), the position ($\mathbf{p}_i$) and fitness of the best point encountered by the particle, and index (g) of the best particle in the swarm. The particle's position represents the current solution to the numerical problem. The particle's next position ($\mathbf{x}'_i$) is determined by the velocity vector:

$$\mathbf{x}'_i = \mathbf{x}_i + \mathbf{v}_i \tag{2}$$

The velocity vector is updated according to the current velocity, the particle's own best position, and the overall best position in the swarm:

$$\mathbf{v}'_i = \chi(w\mathbf{v}_i + \overrightarrow{\varphi_{1i}}(\mathbf{p}_i - \mathbf{x}_i) + \overrightarrow{\varphi_{2i}}(\mathbf{p}_g - \mathbf{x}_i)) \tag{3}$$

where χ is the *constriction coefficient* [7], w is the *inertia weight* [20], and $\mathbf{p}_g$ is the position of the best particle in the swarm. The vectors $\overrightarrow{\varphi_{1i}}$ and $\overrightarrow{\varphi_{2i}}$ are randomly generated for each particle with entries uniformly distributed between 0 and φ_{1max} or φ_{2max}, respectively.

The diversity-guided particle swarm optimizer (DGPSO) was introduced by Vesterstrøm and Riget [23]. It implements the same basic idea as the diversity-guided evolutionary algorithm. Like the DGEA, the DGPSO alternates between phases of exploitation and phases of exploration. Switching between the two phases is controlled by the diversity measure in Eq. 1. The algorithm switches to exploration when the diversity drops below d_{low} and to exploitation when the diversity reaches d_{high}. Diversity is increased by using a different velocity update rule with negative sign on the attraction to the particle's own best position and best position of the entire swarm.

$$\mathbf{v}'_i = \chi(w\mathbf{v}_i - \overrightarrow{\varphi_{1i}}(\mathbf{p}_i - \mathbf{x}_i) - \overrightarrow{\varphi_{2i}}(\mathbf{p}_g - \mathbf{x}_i)) \tag{4}$$

Hence, Eq. 3 is used in exploitation and Eq. 4 in exploration. The thresholds for switching d_{low} and d_{high} were linearly decreased like in the DGEA, and stagnation counting was used to avoid long periods of stagnation.

The parameters for the two particle swarm optimizers are shown in Table 4.

Algorithm	Parameter	Value	Description
STDPSO	ps	20	Population size
	φ_{1max}	2.0	Upper bound for φ_{1i}
	φ_{2max}	2.0	Upper bound for φ_{2i}
	χ	1.0	Constriction coefficient
	w_{max}	0.7	Start value for inertial weight w
	w_{min}	0.3	End value for inertial weight w
	v_{max}	0.15	Maximal velocity in each dimension
DGPSO	ps	20	Population size
	φ_{1max}	2.0	Upper bound for φ_{1i}
	φ_{2max}	2.0	Upper bound for φ_{2i}
	χ	0.65	Constriction coefficient
	w_{max}	0.7	Start value for inertial weight w
	w_{min}	0.3	End value for inertial weight w
	v_{max}	0.15	Maximal velocity in each dimension
	d_{lowmax}	1.0E-5	Start value for d_{low}
	d_{lowmin}	5.0E-12	End value for d_{low}
	$d_{highmax}$	0.1	Start value for d_{high}
	$d_{highmin}$	0.01	End value for d_{high}
	$d_{staglow}$	0.0005	Diversity for stagnation counting
	$d_{stagmax}$	20	Max no. of stagnated generations

Table 4
Parameters for the standard particle swarm optimizer and the diversity-guided particle swarm optimizer.

4 Problem formulation

In parameter identification, the basic idea is to compare the time dependent response of the system and a parameterized model by a performance function giving a measure of how well the model response fits the system response. Normally, the dynamic response of the system is given by the solution to a vector differential equation of the form:

$$\dot{x} = f(\theta, x, u)$$
$$y = g(\theta, x)$$

with the initial condition $x(0) = x_o$. In this system u is the input signal vector, x is the state vector, y is the measurable output vector and θ is the parameter vector. Normally, the system is affected by noise in both states and measurement, which may be real noise or noise caused by unmodeled dynamics. The parameter vector θ is unknown for real systems. Hence, the objective in parameter identification is to determine this vector as accurately

as possible. To do this, a model of the system is introduced with the same structure as the real system. The model is described by:

$$\hat{x} = f\left(\hat{\theta}, \hat{x}, u\right)$$
$$\hat{y} = g\left(\hat{\theta}, \hat{x}\right)$$

with assumed known initial condition $\hat{x}(0) = x_o$. For a given input signal u, an output signal y can be measured from the system; and for a given guess of the parameter $\hat{\theta}$, the output response $\hat{y}$ of the model can be calculated, by imposing the same input signal as for the real system. The system response and the model response can then be compared by a performance function, which in the simple case can be quadratic.

$$I\left(\hat{\theta}\right) = \int_0^T (y - \hat{y})^T \cdot W \cdot (y - \hat{y}) \cdot dt$$

where W is a positive definite weight matrix. The function depends on $\hat{\theta}$ and will obtain its minimum vale zero when $\hat{\theta} = \theta$. The system identification problem can now be formulated as an optimization problem, namely

$$\arg\min_{\hat{\theta}} I\left(\hat{\theta}\right)$$

Obviously, the performance landscape of this problem type may have many local optima and a highly complex topology.

In the next two subsections, the differential equations for the induction motor will be derived together with the performance function.

4.1 Dynamic model of the 1.1 kW induction motor without saturation

The dynamics of the 1.1 kW induction motor can be described by a set of differential equations derived from fundamental laws of physics. Here, only the final equations and not the assumptions will be discussed. A comprehensive derivation of the model was performed by Vas [22]. The motor is supplied by three voltages supported from mains [7] or from an electronic unit that can convert the mains voltages to user specified voltages u_1, u_2, and u_3. In order to simplify the notation, we introduce complex voltages and currents. The transformation of the voltages from a three phase system to a complex system

[7] The national power grid.

is:

$$\begin{aligned} u_s &= u_{sd} + j \cdot u_{sq} \\ &= \frac{2}{3} \cdot (u_1 + a \cdot u_2 + a^* \cdot u_3) \qquad \text{where } a = e^{j\frac{2\pi}{3}} \end{aligned}$$

The real and imaginary voltages are then:

$$\begin{aligned} u_{sd} &= \frac{1}{3} \cdot (2 \cdot u_1 - u_2 - u_3) \\ u_{sq} &= \frac{1}{\sqrt{3}} \cdot (u_2 - u_3) \end{aligned}$$

The inverse transformation from the complex notation to the three phase system is for the currents given by the following set of equations.

$$\begin{aligned} i_1 &= i_{sd} \\ i_2 &= -\frac{1}{2} \cdot i_{sd} + \frac{\sqrt{3}}{2} \cdot i_{sq} \\ i_3 &= -\frac{1}{2} \cdot i_{sd} - \frac{\sqrt{3}}{2} \cdot i_{sq} \end{aligned} \tag{5}$$

From now on we treat the motor as a two phase motor and derive the differential equations in complex notation. The motor consists of a stator where the voltages are applied to two sets of windings perpendicular to each other. In a similar manner, the rotor is composed of two windings, but they can be revolved with respect to the stator. The voltages in the rotor are induced by the movement of the rotor with respect to the stator, the voltage equations for the stator and the rotor are given by the following equations, respectively.

$$\begin{aligned} \dot{\psi}_s &= -R_s \cdot i_s + u_s \qquad && \text{where } i_s = i_{sd} + j \cdot i_{sq} \\ \dot{\psi}_r - j \cdot \omega_r \cdot \psi_r &= -R_r \cdot i_r \qquad && \text{where } i_r = i_{rd} + j \cdot i_{rq} \end{aligned} \tag{6}$$

The left hand sides are the induced voltages and the right hand sides are the supplied voltages reduced by the resistive voltage drop of the windings. The rotor windings do not have any external supply at all, only the stator windings are supplied. ψ_s and ψ_r are the fluxes through the stator and the rotor windings. In the induced rotor voltage equation, ω_r is the speed of the rotor that enters the induced voltage together with the normal induction part due to the flux changes with respect to time. This extra speed dependent term evolves because the rotor terms are calculated into a stator frame of reference. To eliminate the currents in the voltage equations, the flux linkage between the stator and rotor and vice versa have to be calculated. The resulting linkages

are given by the following set of equations.

$$\begin{aligned}\psi_s &= L_{sl} \cdot i_s + L_m \cdot (i_s + i_r) = L_{sl} \cdot i_s + \psi_m \\ \psi_r &= L_{rl} \cdot i_r + L_m \cdot (i_s + i_r) = L_{rl} \cdot i_r + \psi_m\end{aligned} \tag{7}$$

The first expression shows how the flux through the stator windings is composed of the current in the stator itself and by the currents in the rotor windings. The term ψ_m expresses the main flux shared by the stator and the rotor and the rest expresses the leakage flux in the stator and the rotor. The currents in these expressions can be isolated and applied in the differential equations in expression (6). Consequently, the differential equations can be solved by applying the voltage u_s, if the speed ω_r of the motor is known. However, the speed is not an input, but is governed by the following equation of motion for the rotor.

$$\dot{\omega}_r = \frac{1}{J} \cdot (M - M_L) \tag{8}$$

In this expression M is the developed torque of the motor and M_L is the torque load and friction. Furthermore, J is the moment of inertia of the rotor and the load. The developed torque of the motor can be calculated from the electro-mechanical states by the expression:

$$M = \frac{3}{2} \cdot Im\left(\psi_s^* \cdot i_s\right) \tag{9}$$

In our experiments, we assume that the load torque M_L is zero, which eliminates the load torque as an additional input. The model can be described by a set of explicit differential equations if the parameters are constants in the above expressions. In summary, we have:

$$\begin{aligned}\dot{x} &= f\left(\theta, x, u\right) \\ y &= g\left(\theta, x\right)\end{aligned}$$

where, for the actual system the input, state and input are given by the following vectors.

$$\begin{aligned}u &= \left[u_1, u_2, u_3\right]^T \\ x &= \left[\psi_{sd}, \psi_{sq}, \psi_{rd}, \psi_{rq}, \omega_r\right]^T \\ y &= \left[i_1, i_2, i_3, \omega_r\right]^T\end{aligned}$$

and the parameter vector given by:

$$\theta = \left[R_s, R_{r,} L_{sl}, L_{rl}, L_m, J\right]^T$$

Unfortunately, the real world is more complex, because not all the parameters are constant. In real motors, the iron in the motor saturates which means that the main flux ψ_m is a function of the scalar magnetization current $i_m = |i_s + i_r|$. To account for saturation, some rewriting of the above equations is necessary. In equation (7), the currents can be calculated as a function of the stator, the rotor, and the main flux.

$$\begin{aligned} i_s &= \frac{1}{L_{sl}} \cdot (\psi_s - \psi_m) \\ i_r &= \frac{1}{L_{rl}} \cdot (\psi_r - \psi_m) \end{aligned} \tag{10}$$

These currents can be inserted into the voltage equations (6), and thereby eliminate the currents.

$$\begin{aligned} \dot{\psi}_s &= -\frac{R_s}{L_{sl}} \cdot (\psi_s - \psi_m) + u_s \\ \dot{\psi}_r &= -\frac{R_r}{L_{rl}} \cdot (\psi_r - \psi_m) + j \cdot \omega_r \cdot \psi_r \end{aligned} \tag{11}$$

However, this set of differential equations does not express the main flux ψ_m by the states ψ_s and ψ_r. The main flux ψ_m can be expressed by ψ_s and ψ_r by inserting equation (10) into either of following expressions.

$$\begin{aligned} \psi_s &= L_{sl} \cdot i_s + L_m \cdot (i_s + i_r) \\ \psi_r &= L_{rl} \cdot i_r + L_m \cdot (i_s + i_r) \end{aligned} \tag{12}$$

Isolating ψ_m, gives the explicit expression of the main flux:

$$\psi_m = \frac{\left(\frac{1}{L_{sl}} \cdot \psi_s + \frac{1}{L_{rl}} \cdot \psi_r\right)}{\left(\frac{1}{L_m\,(i_m)} + \frac{1}{L_{sl}} + \frac{1}{L_{rl}}\right)} \tag{13}$$

In this expression, the inductance $L_m(i_m)$ is a function of the magnetization current i_m, which can be calculated from the actual motor currents.

$$i_m = \sqrt{(i_{sd} + i_{rd})^2 + (i_{sq} + i_{rq})^2} \tag{14}$$

The varying inductance is determined by magnetization current according to equation (15), which has shown to be a good approximation in practice. L_{mo} is the inductance when the iron in the motor is not saturated and i_{mo} is the

current at which saturation begins. Finally, α is a factor giving the decaying curve shape of L_m at high currents.

$$L_m(i_m) = \begin{cases} L_{mo} & i_m \leq i_{mo} \\ L_{mo} \cdot \left(1 + \alpha \cdot L_{mo} \cdot i_m \cdot \left(\frac{1}{i_{mo}} - \frac{1}{i_m}\right)^2\right)^{-1} & i_m > i_{mo} \end{cases} \tag{15}$$

The differential equation governing the dynamic behavior of the motor is given by equation (11), in which ψ_m is given in an implicit manner by the expressions (13), (14), (10), and (15). The value of ψ_m cannot be explicitly calculated for given values of ψ_s and ψ_r, but has to be approximated iteratively by a fixpoint calculation. The steps in this calculation are first to assume a starting value for L_m and insert this into (13) to calculate ψ_m. Then calculate Eq. (10) to determine i_s and i_r. Next, calculate the scalar magnetization current i_m from (14). Finally, a new value of L_m can be calculated from the lower branch of equation (15). The exact value of ψ_m has been reached if the new L_m value is equal to the L_m value at the beginning of the iteration cycle.

In summary, the differential equations consist of (11) together with (8), and the currents for the output vector is given by (10). By comparison with the model without saturation, the input vector and the output vector are the same, but the structure of the differential equation is different. Thus, the parameter vector is now given by:

$$\theta = [R_s, R_r, L_{sl}, L_{rl}, L_{mo}, i_{mo}, \alpha, J]^T$$

Hence, two extra parameters have been introduced compared with the model of the 1.1 kW motor.

4.3 Formulation of the performance function

The next step is to determine the performance function for the induction motor system. The result of the solution of the differential equations is the states $x = [\psi_{sd}, \psi_{sq}, \psi_{rd}, \psi_{rq}, \omega_r]^T$. From these states, the stator current can be calculated using equation (7). This complex stator current can now be converted to three phase currents comparable with real life currents using (5). The performance function then becomes.

$$I\left(\hat{\theta}\right) = \int_0^T \left((i_1 - \hat{i}_1)^2 + (i_2 - \hat{i}_2)^2 + (i_3 - \hat{i}_3)^2\right) dt$$

Notice that the function does not include the squared deviation in the rotor's motion $(\omega_r - \hat{\omega}_r)^2$. This is excluded because motion measurements on the real

motor are more noisy than the measurements of the currents.

In our experiments, the non-linear differential equations were approximated using the fourth-order Runge-Kutta method [1]. One second of the motor's startup-phase was simulated using a step-size of 0.1 millisecond, i.e., 10000 steps. Hence, we used the sum-of-squared-errors as the performance function (fitness).

$$I'\left(\hat{\theta}\right) = \sum_{t=1}^{10000} \left((i_1(t) - \hat{i}_1(t))^2 + (i_2(t) - \hat{i}_2(t))^2 + (i_3(t) - \hat{i}_3(t))^2\right)$$

5 Experiments and results

The main purpose of the experimentation was to compare the algorithms described in Section 3 with respect to parameter identification of the two induction motors. Each algorithm was tested 20 times on the two motor identification problems. The algorithm parameters listed in Table 1-4 were manually found by trial-and-error tuning, which was done by running a number of preliminary experiments based on previous experience with the algorithms. In this, a known good setting was used as a starting point, which was then refined by adjusting the parameters in the most promising direction based on a few runs.

The experiments reported in this section are carried out using a simulated reference signal, which is generated by the currently best known parameter values of the real motor. The best known parameter values are typically within a 5-10% range of the true, but unknown, values. Parameter estimation at Grundfos is considered highly satisfactory if the percentwise deviation is less than 5% from the true value, which is about the best precision obtainable on the real motor using traditional system identification techniques. The advantage of using a simulated reference signal is that the exact optimum is known, which allows us to compare the algorithms on how close the found solutions are to the known optimum. Thus, a simulated signal will allow us to determine if any of the algorithms should be applied to identification of the real motors.

5.1 Parameter identification of the 1.1 kW motor without saturation

Each algorithm was given 200000 performance evaluations to find the parameters of the 1.1 kW motor. An evaluation takes about 0.05 seconds on a 1.8GHz Pentium 4, and the total time of a run was about 3 hours. In the experiments, a 25 node PC cluster was used to carry out the calculations.

In the 1.1 kW motor model, the two parameters L_{sl} and L_{rl} are linearly dependent and were therefore combined into one parameter. The 5-dimensional search space was discretized to make the search space identical for all algorithms, because the two local search algorithms require discrete search spaces. However, preliminary experiments showed that the PSOs had somewhat lower performance when using the discrete search space. To overcome this problem, the particles positions were kept as continuous variables and rounded just before calculating the performance of a solution. Table 5 lists the reference values, the search intervals, and the step-sizes for the five parameters of the 1.1 kW motor. The ranges and step-size were chosen from practical experience from previous identification of the motor at Grundfos.

	R_s	R_r	$L_{sl} + L_{rl}$	L_m	J
Ref. value	9.203	6.61	0.09718	1.6816	0.00077
Min	6	6	0.029	1.5	0.0001
Max	10	10	0.500	2.0	0.0100
Step	0.0001	0.0001	0.00001	0.0001	0.00001

Table 5
Intervals and step-sizes for the five parameters of the 1.1 kW motor.

The optimization results are shown in Table 6. The table displays the number of runs where the exact optimum was found and the average, best, and worst fitness. As seen in the table, the local search techniques have quite poor performance indicating that the problem is highly multimodal and therefore quite difficult for local search algorithms. However, the relaxed termination criterion implemented in simulated annealing clearly improves the performance. Nevertheless, neither of the two algorithms managed to find the true optimum in any of the 20 test runs. In contrast, the two evolution strategy algorithms had very impressive performance – even the simple algorithm found the true optimum in 16 out of 20 runs. The advanced evolution strategy with adaptation of rotation angles had even better performance. It consistently found a solution close to the true optimum (worst performance 1.20E-4), and in 17 of 20 runs this was the exact optimum. Regarding the generational EAs, the simple EA (SEA) had somewhat intermediate performance. It discovered the true optimum in six runs, but was quite far from it in many cases. Surprisingly, the simple EA is considerably worse than the simple ES. However, further experimentation is necessary to determine whether it is the different selection mechanism, the self-adaptive mutation operator, or a combination of the two that leads to the improved performance. Interestingly, the diversity-guided EA had the best performance of all eight algorithms. The algorithm discovered the true optimum in all 20 runs. To further test the algorithm, an additional 80 runs were performed to see if this was consistent. In total, the true optimum was found in 99 out of the 100 runs. Conclusively, the simple extension implemented in this algorithm significantly improved the performance. Finally, the

particle swarm optimizers also showed good performance. The standard PSO found the true optimum as many times as the simple ES, but the algorithm is clearly not as robust. This is evident from the large average fitness, the standard error, and the worst fitness. The diversity-guided PSO had somewhat better performance compared with the standard PSO's. It has a lower average fitness and find the global optimum in 18 of 20 runs. Again, the standard error and the worst fitness indicate that the algorithm is not as robust as the evolutionary approaches.

Algorithm	# Exact	Avg. fitness	± std. err.	Best fitness	Worst fitness
SDLS	0	141620.61	± 24761.12	45777.88	365286.46
SA	0	34795.91	± 2069.26	13416.14	54441.81
ES1	16	3.91	± 2.13	0.00	26.06
ES2	17	1.80E-5	± 9.82E-6	0.00	1.20E-4
SEA	6	134.70	± 48.82	0.00	933.96
DGEA	20	0.00	± 0.00	0.00	0.00
STDPSO	16	226.69	± 133.87	0.00	2344.47
DGPSO	18	90.60	± 63.65	0.00	1081.88

Table 6
Results for parameter identification of the 1.1 kW motor. Average of 20 runs. The column denoted "# Exact" displays the number of runs where the algorithm discovered the exact optimum.

To further analyze the performance, we calculated the average percentwise deviation from the true value of each variable. Table 7 lists the deviations for the eight algorithms. As shown in the table, the four evolutionary algorithms all had satisfying performance with respect to this criterion, i.e., less than the 5% error defined by Grundfos. In fact, the two evolution strategies and the diversity-guided EA managed to obtain an error of less than 0.2% on all parameters. The two PSOs had good performance, though somewhat worse than the evolutionary algorithms.

The convergence speed is an important aspect of optimization because the fitness evaluation is usually the time-consuming part of real-world optimization. Figure 2 displays the number of evaluations versus the average fitness for the best performing algorithms. The graph show that the ES2 and DGEA had similar convergence speed until about 60000 evaluations, after which the DGEA converges slightly faster.

In summary, the experiments on the 1.1 kW motor show that considerable improvement in performance can be achieved by even rather simple extensions of the standard algorithms. This is the case for all four groups of algorithms. However, the very poor performance of the two local search algorithms underlines the advantage of using a population-based technique.

Algorithm	R_s	R_r	$L_{sl} + L_{rl}$	L_m	J
SDLS	14.49%	23.55%	116.46%	7.56%	83.44%
SA	8.52%	22.57%	166.74%	6.83%	50.06%
ES1	0.13%	0.08%	0.06%	0.11%	0.19%
ES2	0.0002%	0.0002%	0.00%	0.00%	0.00%
SEA	1.29%	0.91%	0.45%	1.01%	2.01%
DGEA	0.00%	0.00%	0.00%	0.00%	0.00%
STDPSO	1.16%	1.39%	0.64%	0.93%	0.97%
DGPSO	0.71%	1.07%	0.13%	0.26%	0.39%

Table 7
Average percentwise deviation for the parameters of the 1.1 kW motor.

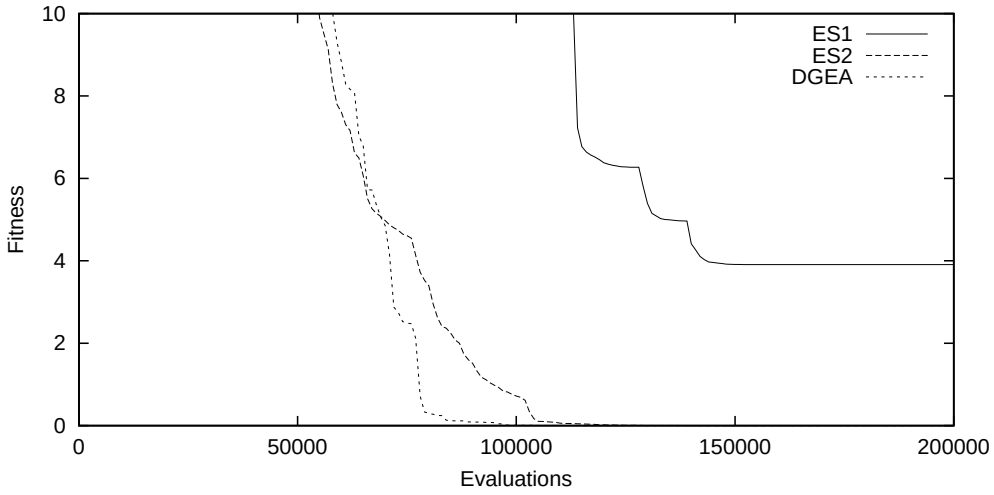


Figure 2. Number of evaluations versus average fitness for the 1.1 kW motor (average of 20 runs).

5.2 Parameter identification of the 5.5 kW motor with saturation

In preliminary runs, the identification of the 5.5 kW motor with saturation turned out to be considerably more challenging than the simpler 1.1 kW motor. Therefore, each algorithm was given 300000 evaluations. For this problem, an evaluation takes about 0.1 seconds, and the total time per run was 6 hours.

In this problem, the two parameters L_{sl} and L_{rl} are independent and cannot be combined. Hence, eight parameters were identified. The 8-dimensional search space was again discretized to suit the local search algorithms. Table 8 displays the reference values, the search intervals, and the step-sizes for the eight parameters of the 5.5 kW motor. Furthermore, preliminary experiments on the DGEA and the DGPSO revealed that setting the parameter d_{lowmax} (Table 3 and 4) to $1.0E-3$ gave better results.

	R_s	R_r	L_{sl}	L_{rl}	L_{mo}	i_{mo}	α	J
Ref. value	3.914	2.71	0.0358	0.0586	1.09	1.096	0.55	0.0084
Min	3.52	1.35	0.03	0.05	0.5	0.5	0.2	0.008
Max	4.30	4.06	0.10	0.10	2.0	2.0	1.0	0.009
Step	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Table 8
Intervals and step-sizes for the eight parameters of the 5.5 kW motor.

The optimization results are listed in Table 9. As mentioned, the problem is significantly more challenging than the 1.1 kW motor's. None of the algorithms found the exact optimum in any run. Regarding the local search algorithms, both SDLS and SA had very poor performance except for one run where SA reached a fitness of about 35. Focusing on the evolution strategies, a surprising result emerged. Apparently, the simple ES outcompeted the advanced ES with respect to the average fitness. This is unexpected because the parameters modeling the saturation (L_{mo}, i_{mo}, and α) appeared to be highly correlated and the advanced ES is particularly designed to handle problems with correlated parameters. A possible explanation is that the correlation is difficult to discover from the fitness landscape. The simple self-adaptation scheme quickly found a mutation variance σ^2 giving reasonable result, but the many variances and rotation angles in the advanced ES may have been too hard to find using the available number of evaluations. Despite of the higher average, the advanced ES actually managed to discover one solution close to the true optimum, which was not the case for the simple ES (best fitness near 5.66). For the generational EAs, the table shows a rather poor performance of the simple EA. The best solution was quite far from the optimum and it only obtained a fitness of about 257.80. Conversely, the diversity-guided EA had by far the best average. The best solution does not quite match the best solution of the advanced ES, but the average fitness and the fitness of the worst solution suggest that the DGEA is more robust. Finally, the particle swarm optimizers had reasonable performance. The standard PSO roughly matches the simple evolution strategy with respect to best found solution. The average performance of both PSOs resemble the performance of the advanced ES. Interestingly, the diversity-guided PSO found the best solution of all algorithms. Again, the standard error and worst fitness indicate that the PSO algorithms may not be as robust as the evolutionary algorithms.

Regarding precision of the found solutions, Table 10 lists the percentwise deviation for the eight parameters. As seen in the table, none of the algorithms matched the 5% performance criterion on all parameters. The diversity-guided EA achieved less than 5% on six out of eight parameters and only slightly worse on the remaining two. Second best is the simple ES, which obtained less than 5% deviation on four parameters and about 5.5% on two parameters. The two

Algorithm	Avg. fitness	± std. err.	Best fitness	Worst fitness
SDLS	11342573.02	± 4908664.64	80315.85	75377366.52
SA	447887.98	± 199274.89	35.43	2867275.03
ES1	82.03	± 13.37	5.66	183.78
ES2	191.88	± 56.31	0.0256	1064.69
SEA	4224.58	± 964.28	257.80	16316.52
DGEA	12.34	± 3.39	0.1215	66.20
STDPSO	182.97	± 88.02	6.53	1441.64
DGPSO	163.25	± 39.73	0.0018	629.45

Table 9
Results for parameter identification of the 5.5 kW motor. Average of 20 runs.

PSOs obtained similar results also having four parameters with an error of less than 5% and two slightly above 5%. As with the 1.1 kW motor, the two local search techniques had rather poor performance.

Algorithm	R_s	R_r	L_{sl}	L_{rl}	L_{mo}	i_{mo}	α	J
SDLS	2.8%	20.3%	51.5%	19.3%	21.5%	21.1%	38.4%	5.8%
SA	2.9%	15.0%	77.4%	19.5%	20.3%	12.0%	33.4%	3.7%
ES1	0.3%	0.3%	10.0%	5.4%	2.6%	5.5%	13.2%	1.2%
ES2	0.1%	0.2%	17.8%	7.8%	5.9%	11.5%	26.8%	0.2%
SEA	0.7%	4.3%	32.1%	27.9%	6.5%	16.4%	42.7%	2.4%
DGEA	0.0%	0.1%	5.2%	2.4%	1.5%	3.1%	6.9%	0.0%
STDPSO	0.3%	0.3%	11.1%	6.3%	2.7%	5.8%	14.7%	1.1%
DGPSO	0.6%	0.4%	7.5%	5.9%	2.8%	5.7%	12.8%	2.1%

Table 10
Average percentwise deviation for the parameters of the 5.5 kW motor.

The convergence speed of the best algorithms is illustrated in Figure 3. The graph shows the average fitness versus number of evaluations for all 20 runs. The simple PSO converged rapidly, but in this case it was passed by the DGEA at about 25000 evaluations and by the simple ES near 150000 evaluations. Interestingly, the advanced ES converged faster than the simple ES.

To sum up, the experimentation on the 5.5 kW motor confirms the results on the 1.1 kW motor. First, simple extensions to the algorithms can have major impact on the performance. This is the case for all four groups of algorithms, although it should be mentioned that the advanced ES had a lower average performance compared with the simple ES's. Despite of this, the advanced ES found better solutions than the simple ES. Second, the local search algorithms had very poor performance in comparison with the performance of the population-based approaches.

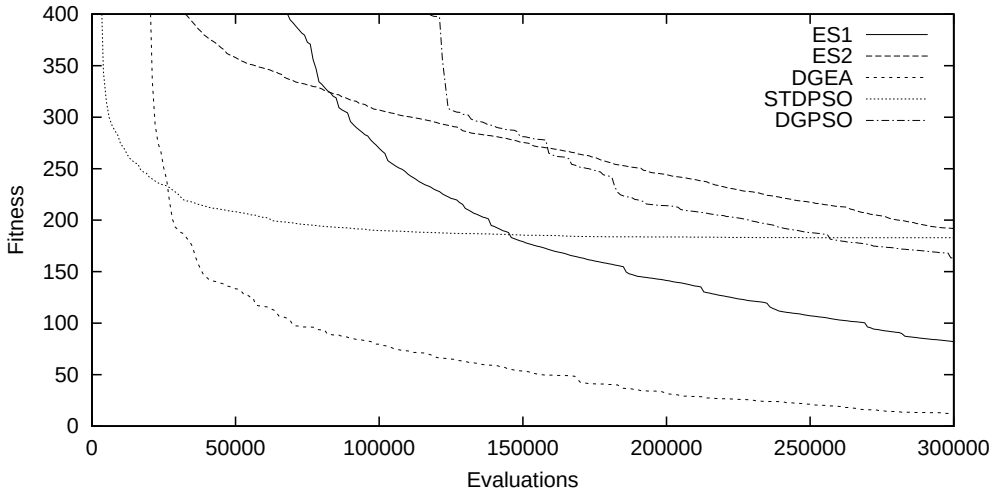


Figure 3. Number of evaluations versus average fitness for the best performing algorithms for the 5.5 kW motor (average of 20 runs).

6 Discussion and conclusions

In this paper, we have compared eight stochastic optimization algorithms with respect to parameter identification of two induction motors. The eight algorithms represent four main groups of stochastic optimization algorithms used today (local search, evolution strategies, generational EAs, and particle swarm optimizers). From each group, we included a simple and an advanced algorithm. Comparing the algorithms, the two local search techniques had the worst performance of all eight algorithms. The simple population-based approaches had rather good performance, while the advanced algorithms had the best performance. Of the advanced techniques, the diversity-guided EA showed the best average performance for both problems, and, for the 1.1 kW motor identification problem, it managed to locate the true optimum in 99 of 100 test runs.

The surprisingly low performance of the local search algorithms indicate that both problems are highly multimodal and therefore difficult to handle with these techniques. To further examine this, an additional 10000 runs of the SDLS in the near vicinity of the global optimum were performed. In these runs, the two search spaces were shinked to ± 10 steps (Table 5 and 8) around the global optimum (reference setting). On both problems, SDLS managed to locate the global optimum in less than 10 out of 10000 runs. Hence, the problems are highly multimodal in the vicinity of the global optimum and possibly also elsewhere. On such problems, the population-based techniques have a clear advantage, since they are capable of generating new candidate solutions far from the existing solutions by exploiting the parallel nature of the population-based search.

Regarding the research gap between applied and basic research, our experiments underline the importance of using more advanced algorithms on real-world problems. Hence, implementing the small extensions of the simple algorithms is indeed worth the effort. In contrast, only the simplest variants of, e.g., evolutionary algorithms are typically used in practical applications. In our view, this is due to three main factors. First, some experience with stochastic optimization is usually required to fully benefit from these techniques. Second, considerable time is often spent on implementing the performance function (e.g. a simulator) leaving limited time for the algorithm. Third, most novel algorithms are tested on artificial benchmark problems having little in common with real-world problems. For practical application, this makes it difficult to find an algorithm that has shown good performance on a similar problem.

In our future work, the next step is to apply the best algorithms to parameter identification using real data obtained from the two motors. The promising results with respect to the 5% deviation criterion (see Section 5) suggest that considerably more precise parameter values can be achieved using the techniques investigated in this paper. However, the results on the 5.5 kW motor still leaves room for improvement. Hence, additional experimentation on this problem may lead to further performance improvements. To this end, the diversity-guided EA seems to be a promising direction for future studies.

Acknowledgements

This work was supported by the Danish Research Council (EVALife project). The authors would like to thank René Thomsen, Mikkel T. Jensen, and Thiemo Krink for comments to early versions of the manuscript.

References

[1] R. Adams, Calculus – a complete course, 3rd Edition, Addison-Wesley Publishers, 1995.

[2] F. Alonge, F. D´Ippolito, G. Ferrante, F. M. Raimondi, Parameter identification of induction motor model using genetic algorithms, Proceedings on Control Theory and Applications 145 (6) (1998), pp. 587–593.

[3] F. Alonge, F. D'Ippolito, F. M. Raimondi, Least squares and genetic algorithms for parameter identification of induction motors, Control Engineering Practice 9 (6) (2001), pp. 647–657.

[4] D. J. Atkinson, P. P. Acarnley, J. W. Finch, Observers for Induction Motor State and Parameter Estimation, IEEE Transactions on Industry Applications,

27 (6) (1991), pp. 1119–1127.

[5] T. Bäck, F. Hoffmeister, H.-P. Schwefel, A survey of evolution strategies, in: R. Belew, L. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman, CA, 1991, pp. 2–9.

[6] T. Bäck, H.-P. Schwefel, Evolution strategies I: Variants and their computational implementation, in: J. Périaux, G. Winter (Eds.), Genetic Algorithms in Engineering and Computer Science, John Wiley & Sons Ltd., 1995, Chapter 6, pp. 111–126.

[7] M. Clerc, The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization, in: P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala (Eds.), Proceedings of the Congress of Evolutionary Computation, Vol. 3, IEEE Press, 1999, pp. 1951–1957.

[8] L. Davis (Ed.), Genetic Algorithms and Simulated Annealing, Research Notes in Artificial Intelligence, Pitman Publishing, 1987.

[9] P. Ju, E. Handschin, Parameter estimation of composite induction motor loads using genetic algorithms, in: Proceedings of the Stockholm Power Technology International Symposium on Electric Power Engineering, Vol. 3, 1995, pp. 97–102.

[10] T. Haque, R. Nolan II, P. Pillay, J. Reynaud, Parameter determination for induction motors, in: Proceedings of the IEEE SOUTHEASTCON'94, 1994, pp. 45–49.

[11] J. H. Holland, Adpatation in Natural and Artificial Systems, University of Michigan Press, 1975.

[12] K. S. Huang, W. Kent, Q. H. Wu, D. R. Turner, Parameter identification for FOC induction motors using genetic algorithms with improved mathematical model, Electric Power Components and Systems 29 (3) (2001), pp. 247–258.

[13] J. Kennedy, R. C. Eberhart, Particle swarm optimization, in: Proceedings of the 1995 IEEE International Conference on Neural Networks, Vol. 4, IEEE Press, 1995, pp. 1942–1948.

[14] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983), pp. 671–680.

[15] L. Ljung, System identification – theory for the user, 2nd Edition, Prentice Hall, 1999.

[16] Z. Michalewicz, D. B. Fogel, How to Solve It: Modern Heuristics, Springer, 2000.

[17] P. Nangsue, P. Pillay, S. Conry, Evolutionary algorithms for induction motor parameter determination, IEEE Transactions on Energy Conversion 14 (3) (1999), pp. 447–453.

[18] A. B. Proca, A. Keyhani, Identification of Variable Frequency Induction Motor Models From Operating Data, IEEE Transactions on Energy Conversion, 17 (1) (2002), pp. 24–31.

[19] I. Rechenberg, Evolutionsstrategie, Vol. 15 of Problemata, Friedrich Frommann Verlag (Günther Holzboog KG), 1973.

[20] Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, in: V. W. Porto, N. Saravanan, D. Waagen, A. E. Eiben (Eds.), Evolutionary Programming VII, Springer, 1998, pp. 591–600, LNCS vol. 1447.

[21] R. K. Ursem, Diversity-guided evolutionary algorithms, in: Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002), Springer, 2002, pp. 462–471.

[22] P. Vas, Electrical Machines and Drives – A Space-Vector Theory Approach, Clarendon Press – Oxford Science Publications, 1992.

[23] J. S. Vesterstrøm, J. Riget, Particle swarms: Extensions for improved local, multi-modal, and dynamic search in numerical optimization, Master's thesis, EVALife, Dept. of Computer Science, Aarhus Universitet (2002).

[24] L. C. Zai, C. L. DeMarco, T. A. Lipo, An Extented Kalman Filter Approach to Rotor Time Constant Measurement in PWM Induction Motor Drives, IEEE Transactions on Industry Applications, 28 (1) (1992), pp. 96–104.