A Data-Driven Optimization Method for Strongly Non-Separable Mixed-Integer Problems

Takahiro Sato
Graduate School of Engineering
Muroran Institute of Technology
Muroran, JAPAN
t-sato@muroran-it.ac.jp

***Abstract*—This paper presents a data-driven optimization method based on tree search-based reinforcement learning to solve strongly non-separable mixed-integer problems. With this method, some variables that mainly affect performance and non-separability are stochastically determined through a modified tree search. Then, other variables are stochastically estimated and optimized by evolutionary algorithms. These probabilities are adapted with a reinforcement learning approach. As a result, the present method makes it possible to easily solve non-separable mixed-integer problems. Moreover, in the reinforcement learning, previous optimization results can be utilized in the next optimizations. The present method is applied to a design problem of an induction motor, which is one of the practical engineering and non-separable mixed-integer optimization problems. It is shown that the present method can solve the design problem under various conditions.**

Keywords—Data-driven, Non-separable, Induction motor, Mixed-integer problem, Reinforcement learning.

I. Introduction

Black-box optimizations, which are usually solved by evolutionary algorithms, is one of the most practical problem classes. We often face such problems in the field of engineering, like in the case of industrial products designs. For example, aerodynamic designs of aircrafts [1], [2] and electric machine designs [3], [4] have successfully been solved by evolutionary algorithms. However, there are still various challenges to address. One of such challenges is to solve strongly non-separable mixed-integer problems [5]-[7]. As an example, let us consider a design problem of a cross-sectional area of an electric motor used in electric vehicles (see Fig. 1). It is known that the motor's output is roughly proportional to the volume of the machine, and the necessary number of coils significantly depends on the input voltage and current. Hence, decision variables such as volume and number of coils play a crucial role in determining the motor characteristics; these variables are a mix of real and integer. In addition, we can see from Fig. 1 that the feasible size of the tooth width (w in Fig. 1) strongly depends on the number of the coils. For this reason, design problems of electric motors are strongly non-separable mixed-integer ones. In many industrial product design problems, like this motor's example, there are some variables that have a strong impact on performance, and their values strongly affect the feasible intervals of other variables. Such main decision variables are here called "fundamental variables." Because of the non-separability due to fundamental variables, it is generally quite difficult to solve such problems with evolutionary algorithms. Hence, typically, some fundamental variables are empirically determined by expert designers in advance, and other decision variables are only optimized.

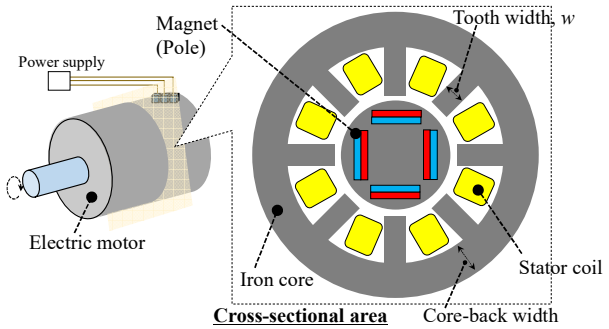


Fig. 1. Practical example of strongly non-separable mixed-integer problem; electric motor design problem.

To solve such strongly non-separable mixed-integer problems, some approaches, in which tree-search-based reinforcement learning and evolutionary algorithms are combined, have been proposed [8], [9]. As for these methods, the decision process of the fundamental variables is modeled as a tree structure based on the knowledge of the problems, and the transient probability is learned from the data. However, in these studies, only the integer fundamental variables are considered. Thus, in this work, we propose a new data-driven optimization algorithm to effectively consider both integer and real fundamental variables by introducing a modified tree structure, and the search path of the modified tree search is learned on the basis of self-generated data. For this reason, the proposed method can relatively easily solve non-separable mixed-integer problems. In addition, it is also possible to immediately solve some optimization problems similar to the previously-solved ones thanks to the utilization of the data.

In this work, the proposed method is first applied to a toy problem with non-separability to clarify the feature of the proposed method. In addition, the proposed method is also applied to a practical engineering design problem featuring an induction motor. It is shown that the proposed method can solve various induction motor design tasks corresponding to various input conditions.

II. Proposed Data-driven Optimization Method

A. Basic concept

Let $\boldsymbol{\chi}$ be decision variables to be optimized. Among $\boldsymbol{\chi}$, real and integer fundamental variables are denoted by $\boldsymbol{z} \in \mathbb{R}^L$ and $\boldsymbol{y} \in \mathbb{N}^M$, respectively, and the other variables are represented with $\boldsymbol{x} \in \mathbb{R}^N$, i.e., $\boldsymbol{\chi}=[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$. The purpose of the optimization

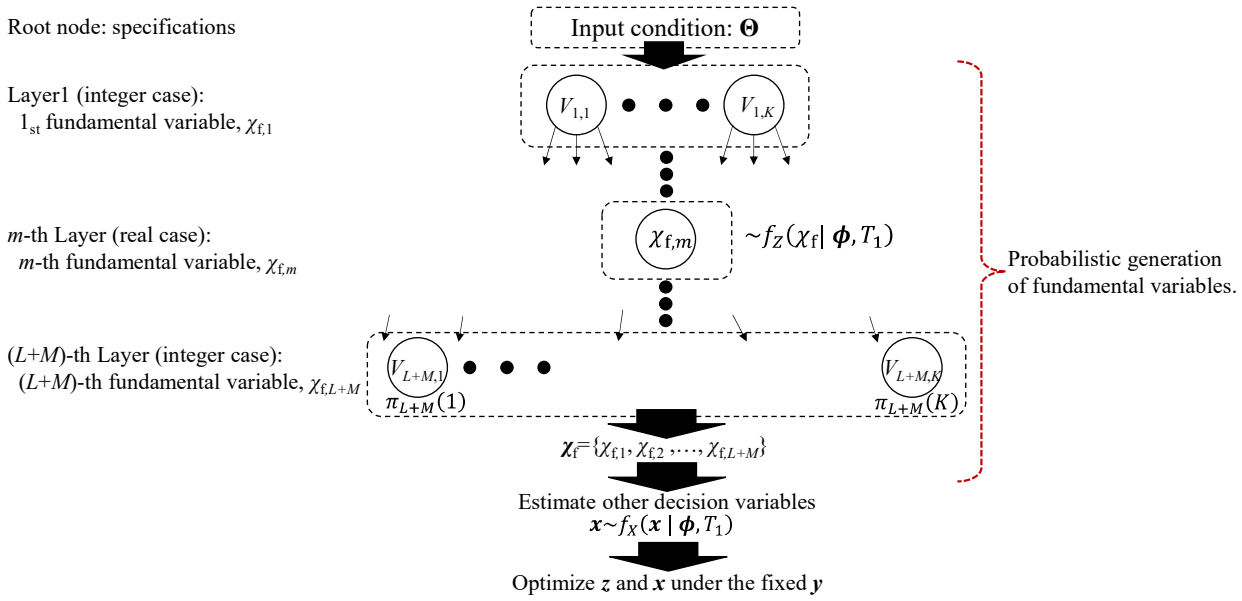


Fig. 2. Extended design search tree

is to find the best $\boldsymbol{\chi}$ by which an objective function, F, for a given condition, $\boldsymbol{\Theta}$, is minimized, that is,

$$\boldsymbol{\chi}_{\text{opt}} = \underset{\boldsymbol{\chi}\in\text{X}}{\text{argmin}}\, F(\boldsymbol{\chi}|\boldsymbol{\Theta}). \tag{1}$$

We here consider that the decision process for determining the fundamental variables in $\boldsymbol{\chi}$ is modeled as a stochastic one to treat the non-separability between fundamental variables. By this assumption, when L=1 and M=2 for example, each fundamental variable is determined:

$$z_1 \sim p(z_1|\boldsymbol{\Theta}), \tag{2}$$

$$y_1 \sim p(y_1|z_1, \boldsymbol{\Theta}), \tag{3}$$

$$y_2 \sim p(y_2|y_1, z_1, \boldsymbol{\Theta}). \tag{4}$$

If these probabilities could be appropriately estimated, the fundamental variables would easily be determined. In some previous works [8], [9], tree structures are used to express the decision process of $\boldsymbol{y}$; however, $\boldsymbol{z}$ is not considered. For this reason, we introduce a modified tree structure to determine $\boldsymbol{z}$ and $\boldsymbol{y}$.

B. Proposed algorithm: EX-DESEAT-DO

Using a tree structure, we propose a new data-driven optimization method combining reinforcement learning for tree structures [10], [11] and evolutionary algorithms. As for this method, the decision process of $\boldsymbol{z}$ and $\boldsymbol{y}$ is modeled as a probability density function and a tree structure. The other variables, $\boldsymbol{x}$, are estimated by a probability density function after determining $\boldsymbol{z}$ and $\boldsymbol{y}$. Then, the determined $\boldsymbol{z}$ and $\boldsymbol{x}$ are optimized under fixed $\boldsymbol{y}$. Consequently, a resultant $\boldsymbol{\chi}$ is obtained. By repeating this generation process, various $\boldsymbol{\chi}$ are obtained and stored. Using the stored $\boldsymbol{\chi}$, the probability density functions and transition probability between the nodes of the tree structure are adapted through a reinforcement learning approach. To realize the above-mentioned process, a modified tree structure, which is here called the "extended design search tree", is proposed as shown in Fig. 2. In addition, the flow of

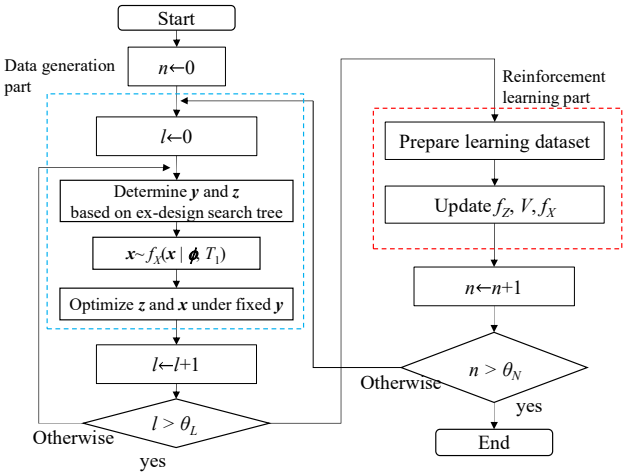


Fig. 3. Optimization flow of EX-DESEART-DO

the proposed algorithm is illustrated in Fig. 3. In this method, first, the extended design search tree is designed on the basis of the knowledge on the problem. Considering the many engineering problems, algorithm users generally have basic knowledge on a target problem. Thus, this method makes it possible to exploit this knowledge through the design of the search tree. This proposed method is here called "EXtended DEsign SEArch Tree-based Data-driven Optimization": EX-DESEAT-DO.

The generation process of $\boldsymbol{z}$ and $\boldsymbol{y}$ is described in detail. Let $\boldsymbol{z}$ and $\boldsymbol{y}$ be collectively denoted by $\boldsymbol{\chi}_\text{f}$, and its m-th component is written as $\chi_{\text{f},m}$. Moreover, the determined m-fundamental variables are denoted by $\overline{\boldsymbol{\chi}}_m$ (note that $\overline{\boldsymbol{\chi}}_0$ is a null vector). First, m is set to 0, and then, the next fundamental variable, $\chi_{\text{f},m+1}$, is generated as follows.

- Fundamental real variables:
  If $\chi_{\text{f},m+1}$ is a real variable, it is generated as:

$$\chi_{\text{f},m+1} \sim f_Z\left(\chi_{\text{f},m+1}|\boldsymbol{\phi}(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}), T_1\right), \tag{5}$$

where f_Z is an arbitrary probability density function, and T_1 is a temperature parameter used to control randomness. Moreover, $\boldsymbol{\phi}(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta})$ is a feature vector derived from $\overline{\boldsymbol{\chi}}_m$ and $\boldsymbol{\Theta}$ (mentioned in section 2.D).

- Fundamental integer variables:

If $\chi_{f,m+1}$ is an integer variable, feasible combinations of $\chi_{f,m+1}$ determined from $\overline{\boldsymbol{\chi}}_m$ are listed as the nodes of the extended design search tree. Let $s_{m,k}$ be the k-th node in the m-th depth. The transition probability, $\pi_m(k)$, to $s_{m,k}$ is given by

$$\pi_{m+1}(k|\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}) = \frac{\exp\left\{\frac{V\left(\boldsymbol{\phi}\left(\overline{\chi}_{k,m+1}, \boldsymbol{\Theta}\right)\right)}{T_2}\right\}}{\sum_{l \in S_{m+1}(\overline{\chi}m)} \exp\left\{\frac{V\left(\boldsymbol{\phi}\left(\overline{\chi}_{l,m+1}\boldsymbol{\Theta}\right)\right)}{T_2}\right\}}, \quad (6)$$

where the value function of $s_{m,k}$ is denoted by $V(\boldsymbol{\phi})$. Note here that $\overline{\boldsymbol{\chi}}_{k,m+1}$ is composed of the previously determined $\overline{\boldsymbol{\chi}}_m$ and a newly determined $\chi_{f,m}$ assigned to $s_{m,k}$. In addition, $S_{m+1}(\overline{\boldsymbol{\chi}}_m)$ is a set of reachable nodes depending on $\overline{\boldsymbol{\chi}}_m$. Moreover, T_2 is a temperature parameter. Depending on $\pi_m(k)$, the next node is probabilistically selected, and $\chi_{f,m+1}$ is determined corresponding to the visited node. By repeating the above-mentioned processes, all the fundamental variables, $\boldsymbol{\chi}_f$, are determined.

After generating $\boldsymbol{z}$ and $\boldsymbol{y}$, the other variables, $\boldsymbol{x}$, are estimated as:

$$\boldsymbol{x} \sim f_X(\boldsymbol{x} \mid \boldsymbol{\phi}(\boldsymbol{\chi}_f, \boldsymbol{\Theta}), T_1), \quad (7)$$

where f_X is an arbitrary probability density function. Finally, the optimization of $\boldsymbol{z}$ and $\boldsymbol{x}$ is performed under the fixed $\boldsymbol{y}$, and the resultant $\boldsymbol{\chi}=[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$ is stored as a candidate solution.

The data-generation process of $\boldsymbol{\chi}$ is repeated and various resultant $\boldsymbol{\chi}$ are obtained. From the stored $\boldsymbol{\chi}$, f_Z, V, and f_X are learned. As a result, one can generate appropriate $\boldsymbol{\chi}$ corresponding to the given $\boldsymbol{\Theta}$. Because it is assumed that strong non-separability appears between $\boldsymbol{z}$ and $\boldsymbol{y}$, the optimization of $\boldsymbol{x}$ and $\boldsymbol{z}$ can be relatively easily performed by ordinary algorithms when $\boldsymbol{y}$ is fixed. The other advantage of EX-DESEAT-DO is that, if we have effective stored data, they can be utilized to learn f_Z, V, f_X. In other words, previous optimization results can effectively be reused.

By the way, the function forms of f_Z, V, f_X are arbitrarily designed, as well the evolutionary algorithm used in EX-DESEART-DO and the feature vector $\boldsymbol{\phi}$. Their implementation in this work is mentioned in section 2.D.

C. Construction of dataset for reinforcement learning

We here explain how to construct the dataset for the learning of f_Z, V, f_X. Let X_D be the set of stored data. From X_D, a dataset, D_m, for the learning of $\chi_{f,m}$ is constructed (m=1,…,L+M). The algorithm is shown in detail in Algorithm 1.

D. Detailed implementations

First, let us consider how to prepare the feature vector $\boldsymbol{\phi}$ from $\overline{\boldsymbol{\chi}}_m$ and $\boldsymbol{\Theta}$. Although there are various approaches, the simplest feature selection approach is adopted:

$$\boldsymbol{\phi}(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}) = [\boldsymbol{a} \odot \overline{\boldsymbol{\chi}}_m, \boldsymbol{b} \odot \boldsymbol{\Theta}], \quad (8)$$

where $\odot$ denotes the Hadamard product, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are user-defined selection vectors whose components are 0 or 1. Of course, the ARD approach [12] would be effective. However, with engineering problems, we have a lot of knowledge on the problems. We consider expressly utilizing this knowledge by determining $\boldsymbol{a}$ and $\boldsymbol{b}$ manually.

Algorithm1. Preparing dataset for reinforcement learning

Input: Stored Data, $X_D=\{\boldsymbol{\chi}_{d,k}=[\boldsymbol{\chi}_f, \boldsymbol{x}, F]_k \mid k=1,2,\ldots\}$

Output: Dataset for each depth, $D_1, D_2, \ldots, D_{L+M}$

1. Count number of different input condition in X_D: N_S
2. Give a input condition label , r_k, to $\boldsymbol{\chi}_{d,k}$: $\boldsymbol{\chi}_{d,k} \rightarrow \boldsymbol{\chi}_{d,k}=[\boldsymbol{\chi}_f, \boldsymbol{x}, r_k]$
3. For each label, apply following processes:
   - A) Initialize extended design search tree. Let $X_{r,k}$ be a set of stored data whose input condition label is r_k.
   - B) For $\forall \boldsymbol{\chi}_{d,k} \in X_{r,k}$, repeat following processes:
     - I. Expand extended design search tree corresponding to $\boldsymbol{\chi}_{d,k}$
     - II. If leaf node for $\boldsymbol{\chi}_{d,k}$ is not expanded, assign $\boldsymbol{\chi}_{d,k}$ to its leaf. Otherwise, if fitness of $\boldsymbol{\chi}_{d,k}$ is better than currently assigned data, $\boldsymbol{\chi}_{d,k}$ is newly assigned. Note that, if the last layer is for real variable, above process is applied to one upper parent layer.
   - C) A dataset, D_{L+M}, for L+M-th layer is composed of all the assigned data
   - D) An index, m, is set to L+M-1, and repeat following process until m=1:
     - I. For each nodes in m-depth, find a child node whose fitness is best among its depth. A data assigned to the best child node is assigned as own data. Note that, if m-th layer is for real variable, above procedure is applied to nodes in m+1-th depth
     - II. A dataset, D_m, for m-th layer is composed of all the data assigned to nodes in m-th depth.
     - III. $m \leftarrow m$-1, go back to step I.
3. Gather obtained D_m for different r_k.
4. For all data in D_m,(m=1,…,L+M), delete data whose $[\boldsymbol{\chi}_f, \boldsymbol{x}]$ is same to the others with better fitness.

Next, we explain the implementations of f_Z, V, f_X. In this work, they are constructed on the basis of Gaussian Process Regression (GPR). To learn V, the input dataset, $D_{in,m}$, and the teacher dataset, $D_{out,m}$, are prepared: $D_{in,m}=\{[\boldsymbol{\chi}_m, \boldsymbol{x}_s]_j \mid j=1,\ldots N_D\}$, $D_{out,m}=\{F_j(\boldsymbol{\chi} \mid \boldsymbol{\Theta}) \mid j=1,\ldots,N_D\}$, where N_D is the number of the data in D_m. Then, we obtain the following values from GPR:

$$\mu_m(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}) = F_{GP1}\left(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta} | D_{\text{in},m}, D_{\text{out},m}\right), \quad (9)$$

$$\sigma_m(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}) = F_{GP2}\left(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta} | D_{\text{in},m}, D_{\text{out},m}\right), \quad (10)$$

where F_{GP1} and F_{GP2} are the mean and variance determined from GPR, respectively. Using them, the node value of $s_{m,k}$ is defined on the basis of the UCB function [13]:

$$V\left(\boldsymbol{\phi}(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta})\right) = \mu_m(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}) + \beta\sigma_m(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}), \quad (11)$$

where β is a hyper-parameter to control the effect of the uncertainty of the regression.

To learn f_Z, the input dataset, $D_{in,m}$ is prepared from m-1-th fundamental variables: $D_{in,m}=\{[\boldsymbol{\chi}_{m-1}, \boldsymbol{x}_s]_j \mid j=1,\ldots N_D\}$. The teachers are the m-th fundamental variable: $D_{out,m}=\{\chi_{f,m+1} \mid j=1,\ldots,N_D\}$. Then, the mean and variance are obtained from (8) and (9), and the probability density function is defined as:

$$f_Z\left(\chi_{f,m+1} | \boldsymbol{\phi}, T_1\right) = N\left(\mu_m(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta}), T_1\sigma_m(\overline{\boldsymbol{\chi}}_m, \boldsymbol{\Theta})\right). \quad (12)$$

Namely, a normal distribution determined from GPR is used, though variance is controlled by T_1. The function f_X is learned in the same way, that is, teachers are: $D_{out,m}=\{\boldsymbol{x}_j \mid j=1,\ldots,N_D\}$. Then, the probability density function is defined from (12). Note that, although the output of f_X is a vector, GPR is simply applied to each component of $\boldsymbol{x}$.

As an optimization algorithm for $\boldsymbol{x}$ and $\boldsymbol{z}$, CMA-ES [14] is used. When starting CMA-ES, the mean vector of the normal distribution in CMA-ES is initialized as the generated $\boldsymbol{x}$ and $\boldsymbol{z}$. To control the search area, the following box-constraints are imposed:

$$\max[(1-\delta)\boldsymbol{x}_0, \boldsymbol{\varepsilon}] \leq \boldsymbol{x} \leq (1+\delta)\boldsymbol{x}_0, \quad (13)$$

where $\boldsymbol{x}_0$ is a predicted $\boldsymbol{x}$ in the generation, and δ is a hyper-parameter. Note that the above is also adopted for $\boldsymbol{z}$.

III. Toy Problem Optimization

A. Problem definition

The performance of EX-DESEART-DO is evaluated through a simple toy problem. In this problem, the dimension of $\boldsymbol{y}$ and $\boldsymbol{z}$ is 1; y and z. Moreover, $\boldsymbol{\Theta}$ is set to ρ. Then, the following objective function is defined:

$$f(\boldsymbol{\chi}|\rho) = (\rho - zy)^2 + g(x,y) + 10^{-6}z, \quad (14)$$

$$g(x,y) = \sum_{i=1}^{N}(x_i - 0.5y)^2, \quad (15)$$

subject to

$$z > 1, \quad (16)$$

and the possible values of y are assumed to be 1, 10, or 20. This problem means that the appropriate y and minimum z for realizing a given ρ must be first found, and simultaneously, the optimal $\boldsymbol{x}$ corresponding to y must also be found. For instance, in the case of ρ<10, y must be 1 because z must be larger than 1. Thus, under y=1, the optimal z and $\boldsymbol{x}$ must be searched for. For example, when considering three different ρ, global optimum solutions are obviously as follows.

- ρ=15: z=1.5, y=10, $\boldsymbol{x}$=(5,…,5)$^{\mathrm{T}}$, f=1.5×10^{-6}
- ρ=20.1: z=1.005, y=20, $\boldsymbol{x}$=(10,…,10)$^{\mathrm{T}}$, f=1.005×10^{-6}
- ρ=25: z=1.25, y=20, $\boldsymbol{x}$=(10,…,10)$^{\mathrm{T}}$, f=1.25×10^{-6}

We check whether the above optimum solutions are obtained by EX-DESEART-DO. The extended design search tree for this problem is defined as Fig. 4. The selection vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$, are set to vectors whose components are all 1.

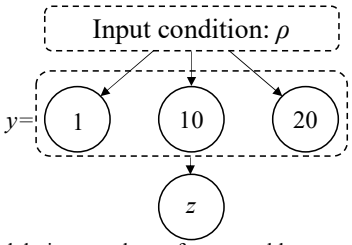


Fig. 4. Extended design search tree for toy problem.

TABLE I. Hyper-parameter settings

parameter	value		
Num. of data-gen. θ_L	1		
Num. of iteration: θ_N	30		
	n>10	10≤n≤20	30<n
Temp. parameter: T_1	1	1	0.1
Temp. parameter: T_2	1000	1	10^{-6}
UCB's parameter: β	+1	0	-1
Parameter interval: δ	10	1	0.5

B. Optimization results

The optimizations are performed 10 times with different random series. The dimension of $\boldsymbol{x}$ is set to 10. The settings of EX-DESEART-DO is summarized in Table I. As for CMA-ES, the number of candidate solutions is set to 20, and CMA-ES is performed over 100 iterations.

The changes in the average objective function of 10 results against the number of function evaluations for two different ρ are shown in Fig. 5. For comparison, optimization results obtained only by CMA-ES with Margin (CMA-ESwM) [15], which is a variant of CMA-ES for mixed-integer problems, are shown. From these two results, it is clearly shown that the conventional CMA-ESwM cannot find the optimum solution in spite of the simple problem, because of the strong non-separability. In comparison, EX-DESEART-DO can find the optimum solution in all the 10 optimizations. This result shows that EX-DESEART-DO can effectively solve strongly non-separable problems.

C. Additional optimization by utilizing stored data

As mentioned in section 2.B, the various data obtained in past optimizations can be utilized in EX-DEART-DO to construct f_Z, V, f_X. Thus, all the data obtained in the optimizations in the case of ρ=15 and 25 are stored, and a new optimization in which ρ is set to 20.1 is performed.

The optimizations are performed in three settings: (A) only generating $\boldsymbol{\chi}$ according to the extended design search tree (CMA-ES is not applied), (B) generating $\boldsymbol{\chi}$ and applying CMA-ES to optimize $\boldsymbol{z}$ and $\boldsymbol{x}$, and (C) just applying the EX-DESEART-DO algorithm from an empty dataset. The results of the three are summarized in Table II, from which we can see that the appropriate y are obtained for (A), though z is not appropriate. This is because the effective y can be predicted

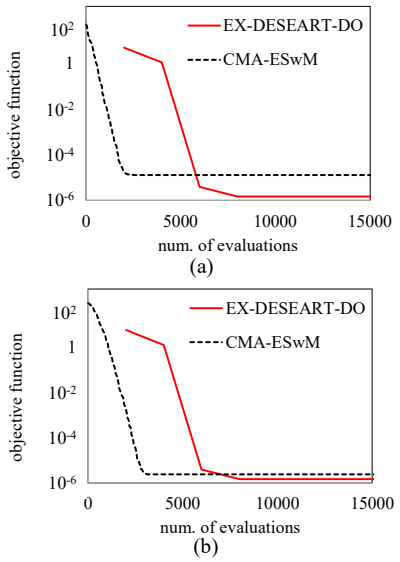


Fig. 5. Toy problem optimization results, (a) ρ=15, (b) ρ=25.

TABLE II. Resultant solutions obtained in case of ρ=20.1

	Case A	Case B	Case C
f	13712	1.005×10^6	1.005×10^6
z	6.85	1.005	1.005
y	20	20	20
average $\boldsymbol{x}$	8.1	10.0	10.0

because the given ρ is between the previous optimization conditions. Hence, in (B), the optimum solution can be obtained only by CMA-ES without additional tree search. From this result, it is revealed that one can effectively obtain better solutions by utilizing stored data in comparison with zero-start optimizations.

IV. Practical Problem-Induction Motor Design

A. Problem definition

As a more practical optimization problem, EX-DESEART-DO is applied to a design optimization problem featuring a squirrel-cage induction motor, which is one of the typical ac motors used in various industrial products. Figure 6 shows a diagram of the sizes and integer values as decision variables. The 5 fundamental integer variables, $\boldsymbol{y}$, are the number of poles, n_P(not appearing in Fig. 6), numbers of stator and rotor slots, N_{st} and N_{rt}, number of parallel circuit, n_Y(not appearing in Fig. 6), and the stator's coil pitch, #n. The real fundamental variables, $\boldsymbol{z}$, are r_{in} and h because the output is assumed to be proportional to the volume. The other 10 variables shown in Fig. 6 are $\boldsymbol{x}$. Namely, there are 5 integer and 12 real variables. In this work, the input condition is set as:

$$\boldsymbol{\Theta} = [P_{\mathrm{T}}, V_{\mathrm{line}}, f_{\mathrm{Hz}}, \Omega_{\mathrm{T}}]^{\mathrm{T}}, \tag{17}$$

where P_{T} is the target output. Moreover, V_{line} is the input line voltage, and f_{Hz} is its frequency. The synchronous speed is denoted by Ω_{T}. As a design optimization, we define a problem to minimize the motor weight by satisfying the given target output, speed, and other constraints as:

$$f(\boldsymbol{\chi}|\boldsymbol{\Theta}) = W \rightarrow min., \tag{18}$$

subject to

$$\eta > \eta_\theta, \tag{19}$$

$$\tau_{\mathrm{ini}} > \tau, \tag{20}$$

$$\frac{\tau_{\mathrm{max}}}{\tau} \geq \tau_\theta, \tag{21}$$

$$s_{\mathrm{max}} < s_{\mathrm{T}}, \tag{22}$$

$$T_{\mathrm{st}} < T_\theta, \tag{23}$$

$$B_{\mathrm{teeth}} \leq B_\theta, \tag{24}$$

$$B_{\mathrm{core}} \leq B_\theta, \tag{25}$$

$$B_{\mathrm{teeth2}} \leq B_\theta, \tag{26}$$

where W denotes the motor weight, η and τ are the efficiency and the torque at the rated condition, respectively, and s_{T} is the rated slip so that the output becomes P_{T} (note here that, if the appropriate s_{T} is not found, the generated data is treated as an unfeasible solution). In addition, τ_{ini} is the initial torque when the slip is 100%, τ_{max} is the maximum torque, and s_{max} is its slip. The temperature of the stator coil is denoted by T_{st}. Moreover, B_{teeth}, B_{core}, and B_{teeth2} are the magnetic flux density at the stator tooth, stator core back, and rotor tooth,

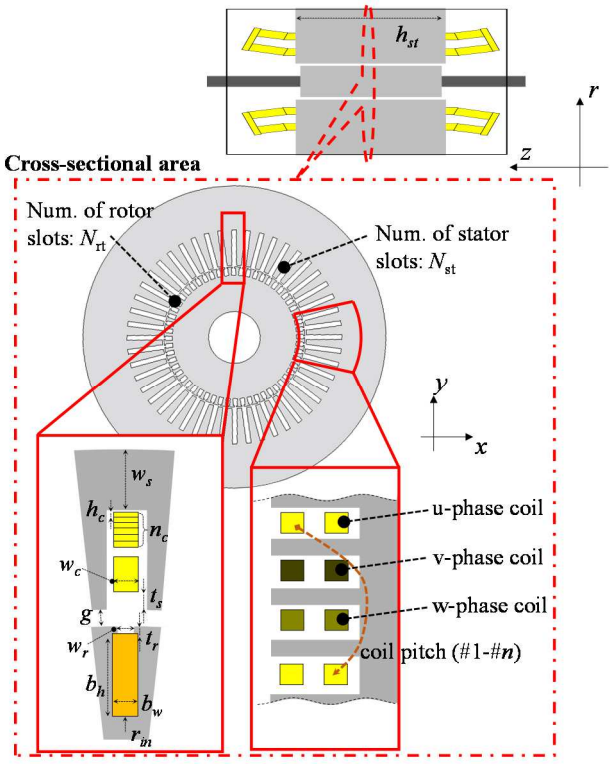


Fig. 6. Squirrel-cage induction motor design problem.

respectively. Then, η_θ=70%, τ_θ=1.5, T_θ=120degC, and B_θ=1.6T are the thresholds. Note here that these characteristic values of the induction motor can be analytically evaluated by the equivalent circuit method. Please see [16] for details.

To consider the constraints in EX-DESEART-DO, f is modified by the simple death-penalty approach as:

$$f(\boldsymbol{\chi}|\boldsymbol{\Theta}) \leftarrow W + \alpha G_{\mathrm{res}}(\boldsymbol{\chi}), \tag{27}$$

where α=10^6 is the penalty parameter, and G_{res} is a residual function of violated constraints. Namely, for example, when (19) is violated, the degree of violation is summed as:

$$G_{\mathrm{res}}(\boldsymbol{\chi}) \leftarrow G_{\mathrm{res}}(\boldsymbol{\chi}) + \left(1 - \frac{\eta}{\eta_\theta}\right)^2, \tag{28}$$

and the other constraints are the same. As for CMA-ES, constraints are considered on the basis of the total ranking approach [17], that is, the fitness, F_m, of the m-th best candidate solutions in CMA-ES is evaluated as:

$$F_m = r_{O,m} + \alpha_r r_{C,m}, \tag{29}$$

$$r_{O,m} = \sum_{i=1}^{M} 1_{\{f_i < f_m\}}, \tag{30}$$

$$r_{C,m} = \sum_{i=1}^{M} 1_{\{G_{\mathrm{res},i} < G_{\mathrm{res},m}\}}, \tag{31}$$

where M is the number of candidates. In short, $r_{O,m}$ and $r_{C,m}$ are the ranking of the objective and residual functions among the candidate solutions, respectively. In addition, α_r is a hyper-parameter, which is here empirically set to M/2.

The extended design search tree for the induction motor problem is defined. First, from the basic theory of motor design that the output is proportional to the volume, $\boldsymbol{z}$ is set to the first depth. Then, the corresponding variable for the

following depth is determined in order of the impact on the following variables. The extended design search tree is shown in Fig . 7. The feasible numbers of $\boldsymbol{y}$ are defined as:

- Number of poles, n_P) automatically determined from motor theory: $n_P=120f_{Hz}/\Omega_T$.
- Number of parallel circuits, n_Y) 1, 2.
- Number of stator slots, N_{st}) integer values for which $N_{st}/(3n_P)$ is 1, 2, 3, 4.
- Number of rotor slots, N_{rt}) integer values for which N_{st}/N_{rt} is 0.7~1.3.
- Coil pitch, #n) integer values for which (#n-1)/($3N_{st}$) is 0.7~1.0.

The selection vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$, for constructing the feature vector are defined on the basis of knowledge on motor designs as:

- Depth 1) $\boldsymbol{b}=[1, 0, 0, 0]^T$
- Depth 2) $\boldsymbol{a}=[0, 0]^T$, $\boldsymbol{b}=[0, 0, 1, 1]^T$
- Depth 3) $\boldsymbol{a}=[0, 0, 0]^T$, $\boldsymbol{b}=[1, 1, 0, 0]^T$
- Depth 4) $\boldsymbol{a}=[1, 0, 0, 1]^T$, $\boldsymbol{b}=[1, 0, 0, 0]^T$
- Depth 5) $\boldsymbol{a}=[0, 0, 1, 0, 1]^T$, $\boldsymbol{b}=[0, 0, 0, 0]^T$
- Depth 6) $\boldsymbol{a}=[0, 0, 0, 1, 1, 0]^T$, $\boldsymbol{b}=[1, 0, 0, 0]^T$
- for $\boldsymbol{x}$) $\boldsymbol{a}=[0, 0, 0, 0, 1, 1, 0]^T$, $\boldsymbol{b}=[1, 0, 0, 0]^T$

where the order of $\boldsymbol{\chi}_f$ and $\boldsymbol{\Theta}$ are $[r_{in}, h_{st}, n_P, n_Y, N_{st}, N_{rt}, \#n]^T$ and $[P_T, V_{line}, f_{Hz}, \Omega_T]^T$, respectively. Of course, any other definitions can be valid depending on the user's knowledge. If you do not have any knowledge, using unit vectors is one of the simplest ways. Instead, the generalization capability may deteriorate when utilizing stored data because the combination input to GPR increases.

B. *Optimization results*

As the input conditions, we consider the following two settings:

- Case 1: P_T=15kW, V_{line}=200V, f_{Hz}=50Hz,Ω_T=1500min^{-1}
- Case 2: P_T=25kW, V_{line}=200V, f_{Hz}=50Hz,Ω_T=1500min^{-1}

The settings of EX-DESEART-DO are summarized in Table III. As for CMA-ES, the number of candidate solutions is set to 50, and CMA-ES is performed over 500 iterations. Note that it takes about a few hours in this setting, because the motor evaluation method is computationally light.

Figure 8 shows the changes in the objective function against the learning iteration for the two cases. We can see that the objective function improves in the early stage of optimization and is remained unchanged between about 50 and 100 iterations. This is because unvisited nodes are mainly searched for due to the larger T_1 and T_2 settings. Then, the objective functions gradually improve after 120 iterations, especially in Case 2. This means that the algorithm tries to greedily search for hopeful nodes because T_1, T_2, and β are set for greedy search. The resultant solutions are summarized in Table IV, which shows that all the constraints are satisfied. The cross-sectional area of the motors for the two resultant solutions are shown in Fig. 9, from which it can be seen that engineeringly valid shapes are obtained at least, i.e., parts that are too thin parts are not obtained. From these results, we can conclude that engineeringly feasible results can be found by EX-DESEART-DO for the given input conditions.

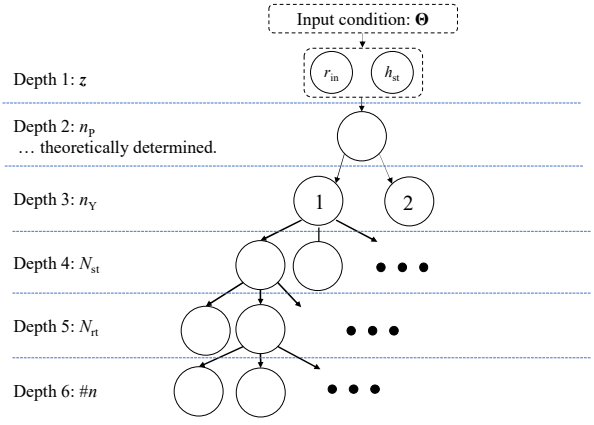


Fig. 7. Extended design search tree for induction motor design

TABLE III. HYPER-PARAMETER SETTINGS FOR INDUCTION MOTOR DESIGN

parameter	value		
Num. of data-gen. θ_L	10		
Num. of iterations: θ_N	150		
	n>60	60≤n≤120	120<n
Temp. parameter: T_1	1	1	0.1
Temp. parameter: T_2	1000	1	10^{-6}
UCB's parameter: β	+1	0	-1
Parameter interval: δ	10	1	0.5

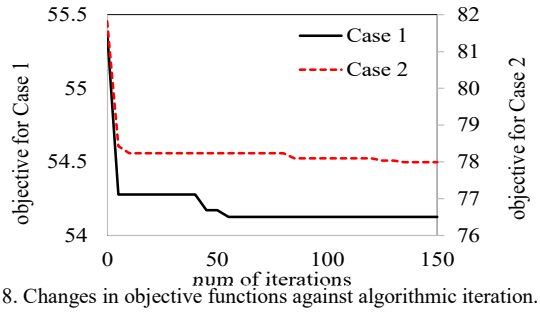
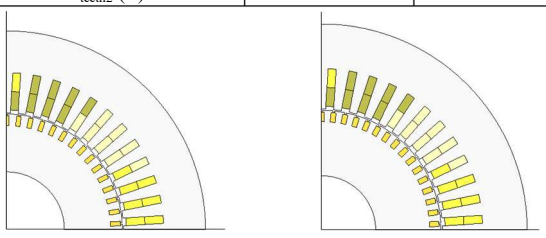


Fig. 8. Changes in objective functions against algorithmic iteration.

TABLE IV. CHARACTERISTICS OF RESULTANT SOLUTIONS

characteristics	value	
	Case1	Case2
$W+\alpha G_{res}$ (-)	54.1	78.0
Output, P_T (kW)	15.01	25.02
Weight, W (kg)	54.1	78.0
Efficiency, η (%)	81.75	84.43
Torque, τ (Nm)	102.8	169.8
Initial torque, τ_{ini} (Nm)	235.6	392.2
Torque ratio, τ_{max}/τ (-)	2.74	2.94
Rated slip s_T, (-)	7.0	6.2
Slip for τ_{max}, s_{max} (-)	53.7	48.6
Coil temperature, T_{st} (degC)	118.6	119.0
Flux density at teeth, B_{teeth} (T)	1.48	1.53
Flux density at coreback, B_{core} (T)	1.58	1.58
Flux density at rotor teeth, B_{teeth2} (T)	1.18	1.35



(a) Case 1 (b) Case 2

Fig. 9. Resultant cross-sectional area of induction motor.

C. Additional optimization by utilizing stored data

For this problem, other optimizations are also performed by utilizing the stored data, that is, all the candidate solutions generated in Cases 1 and 2 were stored. Then, the following two new conditions are set:

- Case 3: P_T=20kW, V_{line}=200V, f_{Hz}=50Hz,Ω_T=1500min^{-1}
- Case 4: P_T=10kW, V_{line}=200V, f_{Hz}=50Hz,Ω_T=1000min^{-1}

The optimization settings are the same as summarized in Table III. Like section 3.C, three types of optimizations are performed in the two cases: (A) only generating $\boldsymbol{\chi}$ according to the extended design search tree (CMA-ES is not applied), (B) generating $\boldsymbol{\chi}$ and applying CMA-ES to optimize $\boldsymbol{z}$ and $\boldsymbol{x}$, and (C) applying the entire process of EX-DESEART-DO. It is expected that acceptable solutions would be obtained for (A) and (B) in the case of Case 3 because the conditions of Case 3 is similar to those of Cases 1 and 2, whereas acceptable solutions would be obtained only by (C) in the case of Case 4 due to the fact that Case 4 has unknown conditions; Ω_T is not equal to the other cases.

The resultant solutions obtained in each case are summarized in Tables V and VI. Considering the above-mentioned assumptions, we can see that the resultant performances of (B) and (C) are almost equivalent in Case 3. In (A) of Case 3, although the values of the fundamental variables are almost equal to those obtained in (B) and (C), some constraint conditions are not satisfied, because not all components of $\boldsymbol{x}$ are correctly estimated. From this result, it can be seen that we can obtain reasonable fundamental variables by utilizing stored data when the input condition is similar to the previously-solved problems. In addition, acceptable results can be obtained only by applying the optimization of $\boldsymbol{x}$ and $\boldsymbol{z}$ without the additional tree search. On the other hand, in Case 4, the performance of the results obtained in (A) and (B) clearly deteriorate in comparison with that obtained in (C). It can be seen from Table VI that this is because the appropriate N_{st} is not generated from the current stored data for the new Ω_T. From this result, it is clear that additional data generation based on additional tree search and learning is necessary when new input conditions are given.

Next, we test the performance of EX-DESEAT-DO for a modified optimization with the same input condition. Thus, (18) and (19) are modified as follows:

$$f(\boldsymbol{\chi}|\boldsymbol{\Theta}) = -\eta \rightarrow min., \tag{32}$$

subject to

$$W < W_\theta, \tag{33}$$

where W_θ is a new threshold. The other constraints are the same. This means that the purpose is changed to maximize the efficiency under the condition that the weight of the motor is smaller than a certain threshold.

For this modification, Case 1 is again optimized for the three optimization types mentioned above. In this optimization, W_θ is set to $1.1W_1 \approx 58$ kg, where W_1 is the resultant weight summarized in Table IV. The resultant solutions are summarized in Tables VII, which shows that the performances for the three types of optimizations are almost equal. This would be because acceptable candidate solutions for the modification were generated when solving the original problem of Case 1. Hence, through the learning of the stored data, effective solutions are immediately generated without additional optimization and tree search.

From these additional numerical results, we can conclude that EX-DESEART-DO can effectively utilize the previous optimization results in an engineeringly practical problem. In particular, when similar conditions are given, acceptable results can immediately be generated only with a data generation and optimization. Otherwise, one can adapt the new conditions by repeating data generation and learning.

TABLE V. CHARACTERISTICS OF RESULTANT SOLUTIONS FOR CASE 3

values of y, z	value		
	(A)	(B)	(C)
r_{rt} (mm)	70	79	75
h_{st} (mm)	113	91	123
n_P	4	4	4
n_Y	1	1	1
N_{st}	48	48	48
N_{rt}	58	58	58
$\#n$	12	12	12
characteristics			
$W+\alpha G_{res}$ (-)	2.1×10^6	69.04	67.23
Output, P_T (kW)	20.25	20.02	20.07
Weight, W (kg)	72.68	69.04	67.23
Efficiency, η (%)	84.44	84.27	83.75
Torque, τ (Nm)	135.0	135.27	135.6
Initial torque, τ_{ini} (Nm)	145.3	223.1	356.1
Torque ratio, τ_{max}/τ (-)	2.13	2.47	3.18
Rated slip s_T, (-)	4.5	5.8	5.8
Slip for τ_{max}, s_{max} (-)	25.6	39.0	51.4
Coil temperature, T_{st} (degC)	135.02	118.5	118.5
B_{teeth} (T)	1.31	1.43	1.50
B_{core} (T)	1.29	1.55	1.58
B_{teeth2} (T)	26.4	1.28	1.16

TABLE VI. CHARACTERISTICS OF RESULTANT SOLUTIONS FOR CASE 4

values of y, z	value		
	(A)	(B)	(C)
r_{in} (mm)	71	72	91
h_{st} (mm)	129	163	77
n_P	6	6	6
n_Y	1	1	1
N_{st}	18	18	72
N_{rt}	20	20	60
$\#n$	4	4	13
characteristics			
$W+\alpha G_{res}$ (-)	5.4×10^6	100.45	50.24
Output, P_T (kW)	10.48	10.04	10.01
Weight, W (kg)	70.47	100.45	50.24
Efficiency, η (%)	55.69	78.00	78.04
Torque, τ (Nm)	100.7	100.9	102.5
Initial torque, τ_{ini} (Nm)	287.7	108.8	177.1
Torque ratio, τ_{max}/τ (-)	9.15	2.13	2.26
Rated slip s_T, (-)	6.0	50.0	6.8
Slip for τ_{max}, s_{max} (-)	18.4	72.3	44.6
Coil temperature, T_{st} (degC)	498.4	116.2	116.6
B_{teeth} (T)	2.64	1.02	1.50
B_{core} (T)	2.95	0.97	1.55
B_{teeth2} (T)	3.57	1.55	1.21

TABLE VII. CHARACTERISTICS OF RESULTANT SOLUTIONS FOR MODIFIED CASE 1

values of y, z	value		
	(A)	(B)	(C)
r_{in} (mm)	64	61	66
h_{st} (mm)	112	112	98
n_P	4	4	4
n_Y	1	1	1
N_{st}	36	36	48
N_{rt}	34	34	64
$\#n$	9	9	12
characteristics			
$W+\alpha G_{res}$ (-)	83.59	85.38	85.83
Output, P_T (kW)	15.02	15.27	15.08
Weight, W (kg)	56.14	57.97	57.74
Efficiency, η (%)	83.59	85.38	85.83
Torque, τ (Nm)	100.8	100.3	98.39
Initial torque, τ_{ini} (Nm)	165.7	124.3	116.0
Torque ratio, τ_{max}/τ (-)	2.58	2.69	2.79
Rated slip s_T, (-)	5.1	3.1	2.4
Slip for τ_{max}, s_{max} (-)	35.0	22.0	20.4
Coil temperature, T_{st} (degC)	118.5	117.2	118.7
B_{teeth} (T)	1.43	1.43	1.48
B_{core} (T)	1.54	1.54	1.48
B_{teeth2} (T)	1.33	1.45	1.57

V. CONCLUSION

For strongly non-separable mixed-integer optimization problems, a design search tree-based data-driven optimization method, called EX-DESEART-DO, has been proposed. With this method, the decision process of fundamental decision variables, which are assumed to mainly affect objectives, are modeled by the extended design search tree. On the basis of the tree search and optimizations, various optimization results are generated. Using them, probability functions are learned. Because the non-separability due to fundamental variables is effectively considered through the extended tree structure, the EX-DESEART-DO makes it easy to optimize strongly non-separable mixed-integer problems. The EX-DESEART-DO has been applied to the two optimization problems. For the toy problem, it has been shown that the EX-DESEART-DO can effectively solve a non-separable mixed-integer problem, while the conventional CMA-ESwM cannot. It has also been shown that EX-DESEART-DO can solve the practical induction motor optimization problem. Moreover, it has been revealed that EX-DESEART-DO can effectively solve new optimization problems by utilizing previous results.

For future work, we will consider more effective learning methods for constructing f_Z, V, f_X, e.g., we will use other function approximators like deep neural networks.

REFERENCES

[1] C. Fujio and H. Ogawa, "Physical insight into axisymmetric scramjet intake design via multiobjective design optimization using surrogateassisted evolutionary algorithms," *Aerospace Science and Technology*, vol. 113, 2021, art. no 106676.

[2] W. J. Koning, E. A. Romander, W. Johnson, "Optimization of low reynolds number airfoils for martian rotor applications using an evolutionary algorithm," *In AIAA Scitech 2020 Forum*, 2020.

[3] T. P. M. Bazzo, et. al., "Multidisciplinary design optimization of direct-drive PMSG considering the site wind profile," *Electric Power Systems Research*, vol. 141, pp. 467-475, 2016.

[4] H. Chen, C. H. T. Lee, "Parametric Sensitivity Analysis and Design Optimization of an Interior Permanent Magnet Synchronous Motor," *IEEE Access*, vol. 7, pp. 159918-159929, 2019.

[5] A. Agapie, "Estimation of distribution algorithms on non-separable problems," *International Journal of Computer Mathematics*, vol. 87, no. 3, pp. 491-508, 2010.

[6] N. Hansen, et. al., "Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems," *Applied Soft Computing*, vol. 11, no. 8, pp. 5755-5769, 2011.

[7] M. Meselhi, R. Sarker, D. Essam, S. Elsayed, "A decomposition approach for large-scale non-separable optimization problems," *Applied Soft Computing*, vol. 115, 2022, art. no 108168.

[8] T. Sato, M. Fujita, "A Data-Driven Automatic Design Method for Electric Machines Based on Reinforcement Learning and Evolutionary Optimization," *IEEE Access*, vol. 9, pp. 71284–71294, 2021.

[9] H. Sato, H. Igarashi, "Automatic Design of PM Motor Using Monte Carlo Tree Search in Conjunction With Topology Optimization," *IEEE Trans. Magn*., vol. 58, no. 9, 2022, art. no. 7200504.

[10] D. Silver, A. Huang, C. Maddison, et. al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[11] D. Silver, et. al., "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp.1140-1144, 2018, doi: 10.1126/science.aar6404.

[12] K. Liu, Y. Li, X. Hu, M. Lucu, W. D. Widanage, "Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries," *IEEE Transactions on Industrial Informatics*, vol.16, no.6, pp. 3767-3777, 2019.

[13] N. Srinivas, A. Krause, S. Kakade, M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *Proc. of International Conference on Machine Learning*, 2010, doi: 10.1109/TIT.2011.2182033.

[14] N. Hansen, A. Auger, "Principled Design of Continuous Stochastic Search: From Theory to Practice," *Theory and Principled Methods for the Design of Metaheuristics. Natural Computing Series*, Springer, 2014, doi: https://doi.org/10.1007/978-3-642-33206-7_8Y.

[15] R. Hamano, S. Saito, M. Nomura, S. Shirakawa, CMA-ES with Margin: Lower-Bounding Marginal Probability for Mixed-Integer Black-Box Optimization, In Genetic and Evolutionary Computation Conference, 2022.

[16] T. A Lipo, "Introduction to AC Machine Design," *IEEE Press Series on Power Engineering*, 2017.

[17] N. Sakamoto, Y. Akimoto, "Proposal of a Linear Constraint Handling Method for Black-Box Optimization with Invariance Properties," *Transaction*