mathematics

MDPI

Article

Optimal Agent Search Using Surrogate-Assisted Genetic Algorithms

Seung-Soo Shin [1] and Yong-Hyuk Kim [2,*]

[1] Department of Computer Science, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Republic of Korea
[2] School of Software, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Republic of Korea
* Correspondence: yhdfly@kw.ac.kr

Abstract: An intelligent agent is a program that can make decisions or perform a service based on its environment, user input, and experiences. Due to the complexity of its state and action spaces, agents are approximated by deep neural networks (DNNs), and it can be optimized using methods such as deep reinforcement learning and evolution strategies. However, these methods include simulation-based evaluations in the optimization process, and they are inefficient if the simulation cost is high. In this study, we propose surrogate-assisted genetic algorithms (SGAs), whose surrogate models are used in the fitness evaluation of genetic algorithms, and the surrogates also predict cumulative rewards for an agent's DNN parameters. To improve the SGAs, we applied stepwise improvements that included multiple surrogates, data standardization, and sampling with dimensional reduction. We conducted experiments using the proposed SGAs in benchmark environments such as cart-pole balancing and lunar lander, and successfully found optimal solutions and significantly reduced computing time. The computing time was reduced by 38% and 95%, in the cart-pole balancing and lunar lander problems, respectively. For the lunar lander problem, an agent with approximately 4% better quality than that found by a gradient-based method was even found.

Keywords: surrogate-assisted computation; genetic algorithm; agent optimization

MSC: 68T05; 68T20; 68W50

Citation: Shin, S.-S.; Kim, Y.-H. Optimal Agent Search Using Surrogate-Assisted Genetic Algorithms. *Mathematics* **2023**, *11*, 230. https://doi.org/10.3390/math11010230

Academic Editor: Ioannis G. Tsoulos

Received: 21 November 2022
Revised: 29 December 2022
Accepted: 30 December 2022
Published: 2 January 2023

1. Introduction

1.1. Motivation

Intelligent agents are systems used to achieve various goals in uncertain environments [1]. They receive states and rewards from the environment and select actions through policies that constitute their action probability distributions based on the current state. As the complexity of an agent's environment space increases, policies can be approximated as deep neural networks (DNNs) [2]. The parameters of these DNNs are optimized through evolution strategies or the gradient descent algorithm using backpropagation [3]. However, these methods are accompanied by simulation-based evaluation for optimization, which incurs enormous costs when applied to expensive real-world environments. Therefore, to reduce the simulation cost, we propose surrogate-assisted genetic algorithms (SGAs) for optimizing the DNN parameters of agents and compare their computational costs and optimization performance with those of a simulation-based method. The proposed SGAs replace the real fitness function of the genetic algorithm (GA) [4,5] with surrogate models that predict cumulative rewards according to the DNN parameters of the agent.

1.2. Contribution

However, SGAs mislead to a false optimum if the fidelity of surrogates is low [6,7]. To solve this problem, we propose three stepwise improvements to increase the surrogate-assisted efficiency associated with the optimization performance of SGAs. The first is to use multiple surrogates, which can reduce the prediction error by combining various machine

learning-based classifiers and regressors. The second is data standardization to address overfitting and outlier problems caused by the experimental data. The third is random sampling with dimensional reduction to address the problem of high-dimensional and limited samples, wherein a domain is set up through dimensionality reduction and samples are added to it. We conducted experiments on benchmark environments of cart-pole balancing and lunar lander using the proposed SGAs, and compared their performance and computational costs with the gradient-based baseline method for each environment. The contributions of this paper are as follows: we proposed novel SGAs to optimize DNNs of intelligent agents to reduce computational costs compared to the double deep *Q*-network and we presented three stepwise improvements that successfully solve the misleading problems of SGAs.

1.3. Organization

The remainder of this paper is organized as follows. Section 2 presents the related research. In Section 3, the structure and techniques of the proposed SGAs with stepwise improvements for optimizing the DNN parameters of intelligent agents are introduced, and the experimental details are presented. In Section 4, the results obtained in the experiments are presented and discussed. Finally, the conclusions and future research directions are presented in Section 5.

2. Related Work

2.1. Surrogate-Assisted Genetic Algorithm

The GA is a global optimization method that mimics natural selection and genetics. Through the computation of the selection and crossover operators, cooperation is achieved among several solutions, which enables a faster optimization search than simple parallel search methods. The GA is applicable to nonlinear or incalculable problems, regardless of the shape of the evaluation functions. Owing to these advantages, the GA is widely used in various fields such as natural science, engineering, humanities, and social sciences [8–10]. However, it generally requires evaluating all chromosomes in the population at each generation, which is inefficient if the computational cost of the fitness function is very high. To reduce the fitness function cost, SGA studies have replaced the surrogate model with real fitness functions [11–13]. Surrogate-assisted methods have the advantage of reducing evaluation costs, but they can mislead to false optima [14]. To prevent this, a strategy for managing the genetic process or the surrogate model is required. In the genetic process, evolution control involves a hybrid evaluation that evaluates a specific chromosome of a population as the real fitness function [15,16]. Surrogate and surrogated-assisted performances can be degraded by problems such as the curse of dimensionality and lack of environmental data. Degradation of surrogate-assisted performance can be prevented by employing improvement strategies such as multiple surrogates [17] and intelligent data sampling [18].

2.2. Neural Network Optimization of Agents

The learning of intelligent agents used to solve problems is performed by maximizing the cumulative rewards using an unsupervised method [19]. As the problem environment is complex, when the state and action space are represented in high-dimensional space, the agent is approximated by a DNN. Optimizing the DNN parameters of an agent involves two methods. The first is deep reinforcement learning (DRL), wherein the parameters are optimized by computing or approximating the gradient of an agent's DNN [20]. DRL involves directly learning policies, which are the action probability distributions according to the state, or Q-functions, which represent the state value according to the action in the given state [21,22]. The second method optimizes the agent DNN parameters for GA. Unlike DRL, this method can reduce the computation time by configuring a non-gradient method in parallel. Some studies proposed learning both the topologies and DNN parameters, and showed performance improvements over DRL [23,24]. In this study, we

collected experimental points through the learning process of the gradient-based double deep *Q*-network (DDQN) [25] and designed a surrogate model. We searched for the optimal agent DNN using GAs, which can reduce computing time by parallel structure and on which there have been a lot of prior studies [23,24]. Since the collected experimental points are network parameters of a fixed structure, a conventional GA [26] for optimizing the structure is considered to be suitable.

3. Materials and Methods

3.1. SGA

3.1.1. Encoding and Fitness

In GAs, the solutions to problems are encoded in chromosomes. Encoding is designed in various ways, based on the nature of the problem or the GA operator. In this study, we encoded all weights and biases of DNNs into length-L vectors of 1D real numbers to represent the agent's DNN (Figure 1). The network θ is an agent DNN and should be satisfied with the range of set Θ of training network in the surrogate model. The first experimental environment, cart-pole balancing, was designed with four architecture cases that were encoded in real numbers with lengths of 23, 35, 37, and 67. The reason for these architecture cases is to minimize the number of parameters encoded. A small number of encoded parameters can avoid the performance degradation of GAs' computing time and surrogates. The second experimental environment, the lunar lander, requires a more complex architecture for problem solving because its action and state space are more complicated than those of cart-pole balancing. Because the lunar lander has 1476 parameters, the surrogate-assisted performance decreases and the GA computing time increases when encoding all of them. To address this, we measured their Gini importance using DNN parameters and encoded the top 100 parameters. Fitness was used to measure the quality of the solution encoded in the chromosome. Because fitness is used as an indicator for selecting good-quality solutions in selection operations, it is important to design a fitness function according to the GA structure and problem characteristics. We evaluated the chromosome quality using a surrogate model that predicted a cumulative reward for the agent's DNN parameters.

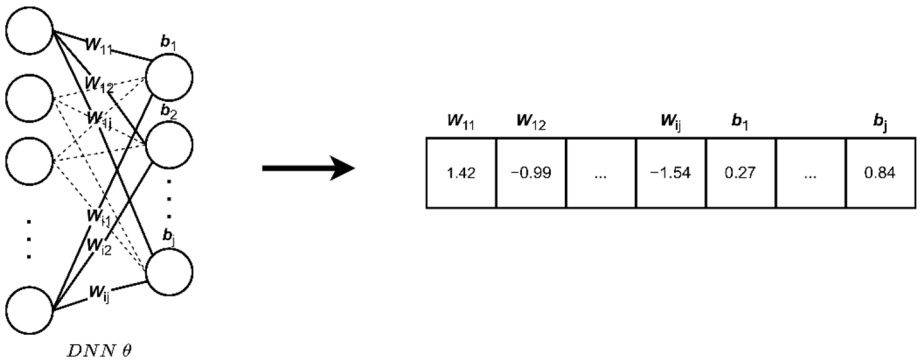


Figure 1. Example of DNN encoding.

3.1.2. Selection and Crossover

The selection operation is used to select the parent chromosomes used for crossover generation, and the probability of selecting high−quality chromosomes should be high. In this study, chromosomes were selected based on the quality measured with surrogate assistance. When a roulette-wheel is directly applied to our SGAs, the fitness range of the initial population is very wide due to the penalties of classifiers in the surrogates, which will be described in Section 3.1.4. Therefore, the diversity of GAs may decrease due to premature convergence. To avoid the problem, we adjusted fitness values to increase the probability that low-quality solutions will be selected at early generations. The roulette-wheel method adjusts the population fitness such that the maximum quality is k times

the minimum quality, and the parent chromosomes are selected according to the adjusted fitness. k is a selection pressure variable that determines convergence, and as the value increases, the probability deviation selected according to fitness increases. Equation (1) calculates the adjusted fitness f_{adj} from the existing fitness f is as follows:

$$f_{adj} = (C_{min} - f) + \frac{C_{min} - C_{max}}{k - 1}, \ldots k > 1, \quad (1)$$

where C_{max} and C_{min} the maximum and minimum in existing fitness.

For populations size P, $2P$ chromosomes for crossover are selected. Then, the time complexity of the selection operator becomes $\Theta(PL)$, where L is the chromosome length. In this study, we adopted the extended-box crossover because the chromosomes were encoded with real numbers [27]. The extended-box crossover extends the maximum and minimum ranges of each gene of the parent chromosomes using the expansion rate α and selects random real numbers from the extended ranges. Additionally, to reflect the learning domain of the surrogate model, the boundary value was set per gene, and the operator was adjusted such that it did not exceed the boundary. Compared to the box crossover, the extended-box crossover prevents early convergence and bias problem. The time complexity of the extended-box crossover is $\Theta(L)$. The pseudocode of the extended-box crossover is shown in Algorithm 1. In this study, the typical selection pressure k and the expansion rate α were used, as 3 and 0.5, respectively.

Algorithm 1 Extended-box Crossover

Extended-box Crossover (*p1*, *p2*)
L:= Chromosome length
for $i \leftarrow 1$ *to* L
$m \leftarrow$ **min**($p1_i$, $p2_i$), $M \leftarrow$ **max**($p1_i$, $p2_i$)
$em \leftarrow m$-$\alpha(M$-$m)$, $eM \leftarrow M$+$\alpha(M$-$m)$
$z_i \leftarrow$ a random real number in [**max**(em, **min**$_{\theta \in \Theta}$ θ_i), **min**(eM, **max**$_{\theta \in \Theta}$ θ_i)]
//Θ: set of training networks
//θ_i: the i-th value of θ encoded by Section 3.1.1
return $Z = (z_1, z_2, \ldots, z_L)$

3.1.3. Mutation and Replacement

The mutation operator adds diversity to the population to reduce chromosomal similarity and prevent convergence to the local optimum. In this study, we generated random values for each gene in the chromosome, compared them with the threshold to replace them with a random real number, and verified that the replaced gene satisfied the domain range. The mutation rate was set to 0.015, and the rate was chosen as showing the best optimization performance in the range [0, 0.05]. Our mutation operator's time complexity is $\Theta(L)$. The pseudocode of the mutation is shown in Algorithm 2. The replacement operator replaces the offspring generated by previous operators with parent chromosomes. In this study, generational GA, which replaces all offspring with parents, and elitism, which preserves the highest-quality chromosomes, were used.

Algorithm 2 Mutation

Mutation (*Chromosome*)
L:= Chromosome length
p:= Mutation rate
for $i \leftarrow 1$ *to* L
if (**random** [0, 1) < *p*)
Chromosome[*i*] $\leftarrow$ **random**[**min**$_{\theta \in \Theta}$ θ_i, **max**$_{\theta \in \Theta}$ θ_i]

3.1.4. Surrogate Model with Stepwise Improvement

The surrogate model performance is not a compulsory evaluation factor of SGA; however, the optimization performance may degrade in the surrogate-assisted method. In this study, three stepwise improvements were applied to improve the surrogate-assisted performance. The first involves using an ensemble with multiple homogeneous surrogates, rather than a single surrogate, to improve the performance. The second involves standardization of the input data for surrogate-assisted training. The third involves adding experimental points other than those in the existing simulation-based methods through data sampling. Sampling that reflects the problem domain can improve the performance of the surrogates. We reduced the dimensions through principal component analysis (PCA) and added experimental points that satisfied the domain. In the following, these three stepwise improvements are described in more detail:

1. Heuristic Measure using Multiple Surrogates: The advantage of using ensembles that employ multiple surrogates is that it is possible to address the degradation problem of a single surrogate, and the predicted performance variance helps avoid false optima [28]. In this study, we designed multiple homogeneous surrogates that combine classifiers and regressors designed using the same training data. The fitness function of the proposed surrogate is expressed as the sum of the predicted values of the regressor and output of the classifiers that reflects the penalty weights. The decision boundary of the classifiers is determined by the cumulative reward quality of an agent based on the problem. The heuristic measure that returns the results for the input network θ is expressed as Equation (2). $c1$ and $c2$ are classifiers for the input network θ, and have an output value of 1 or 0. r is a regressor and outputs the cumulative reward prediction for θ. p_{c1} and p_{c2} are negative penalty constants such that $p_{c1} \ll p_{c2} \ll 0$.

$$\textit{Heuristic measure}(\theta) = \begin{cases} r(\theta)+p_{c1} & \text{if } c1(\theta)=0, \\ r(\theta)+p_{c2} & \text{if } c1(\theta)=1 \text{ and } c2(\theta)=0, \\ r(\theta) & \text{if c1}(\theta)=1 \text{ and c2}(\theta)=1 \end{cases} \quad (2)$$

2. Data Standardization: Because this study collected experimental points through a gradient-based, we obtained numerous low-quality outliers prior to the convergence to the optimal solution. To prevent performance degradation of the surrogate owing to this problem, we calculated the average m and standard deviation σ for each agent's DNN parameter and adjusted the input data to have a standard normal distribution. The standardized value Z for the input random variable X is calculated as follows:

$$Z = \frac{X - m}{\sigma} \quad (3)$$

3. Random Sampling using PCA: Training data used for designing surrogate models for optimizing DNNs of agents are limited and highly dimensional. This could not sufficiently represent the problem domain, which may deteriorate the quality of surrogate-assisted computation. Data sampling was applied to address this problem, reduce biases, and increase robustness to outliers [29]. However, in the case of higher dimensions, the domain is very wide and it is difficult to add samples. In this study, we reduced the dimensions through PCA [30] and added experimental points randomly and uniformly in the reduced domains. PCA analyzes the correlation between DNN parameters and extracts the principal components representing them, thereby minimizing information loss in high-dimensional spaces and performing dimension reduction that reflects the domain. Figure 2 shows an example of data sampling after 2D reduction through PCA, wherein Figure 2a shows a scatter plot of training data before sampling and Figure 2b shows the scatter plot after performing random data sampling through PCA. From Figure 2b, it can be qualitatively confirmed

that the cluster is well-formed according to the decision boundary of the classifier presented in the heuristic measure using multiple surrogates.

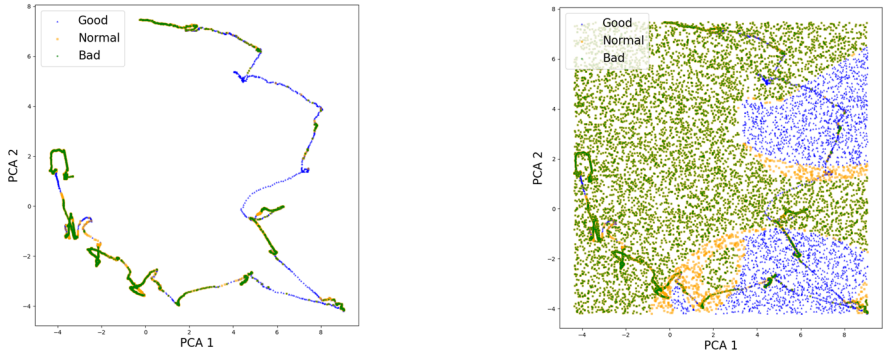


Figure 2. Scatter plots expressed in 2D through PCA: (**a**) training data and (**b**) random sampling added to (**a**).

The flowchart of the proposed SGA is shown in Figure 3. GA operators designed in Section 3 optimize the encoded parameters of an agent DNN, and the fitness values of the chromosomes are evaluated by surrogate model with stepwise improvements of Section 3.1.4.

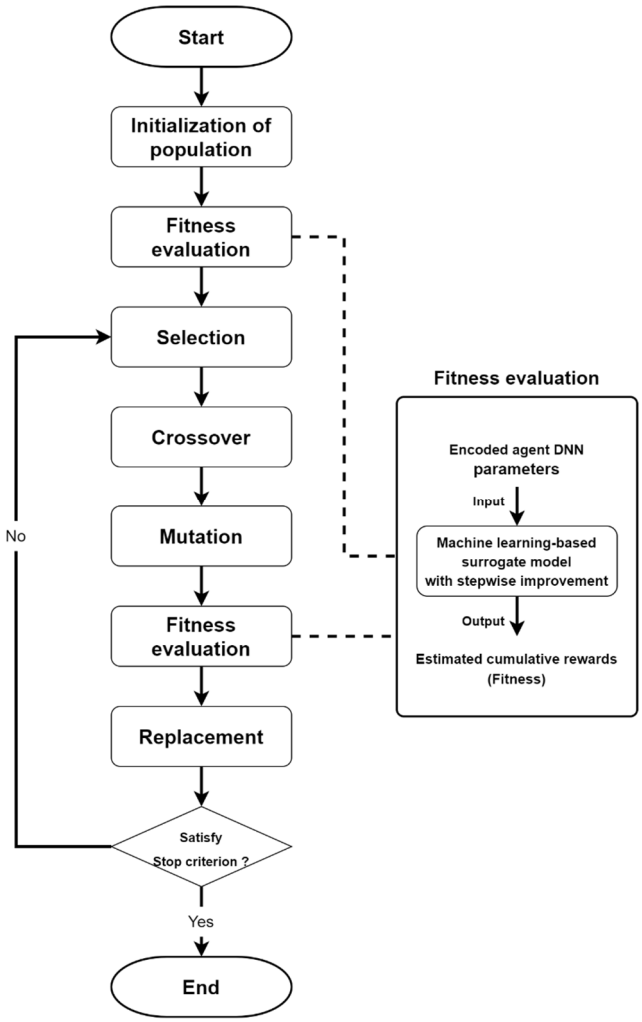


Figure 3. Flowchart of the proposed surrogate-assisted genetic algorithm.

3.2. Experiments

3.2.1. Objective for Optimal Agent Search

The purpose of a general agent is to maximize the cumulative reward according to the environment. An agent's reward function typically takes as input a trajectory, which is a set of sequences of actions and states. The objective function for agent optimization is as follows:

$$\text{Maximize } \sum_{t=1}^{T} Env^R(s_t, \theta(s_t)) + F\left(\prod_{t=1}^{T}\{(s_t, \theta(s_t))\}\right), \tag{4}$$

where $s_{t+1} = Env^N(s_t, \theta(s_t))$ and $Env^R(s_u, \theta(s_u)) = 0$ for $u \geq t$ if s_t is not feasible. θ is an agent DNN, returning an action vector that maximizes the cumulative rewards when s_t is an input. The output of Env^R is a loss of the reward calculated through the input of agent's actions. Env^N takes a pair of action and state as input and returns the next state vector at next time step. T is the maximum time step value. F returns the agent's reward.

3.2.2. Cart-Pole Balancing Problem

In this study, we experimented using cart-pole balancing, a classical continuous control environment that is mainly used as a benchmark in DRL and neuroevolution. Cart-pole balancing comprises a pole l attached through a pivot joint on cart M, as shown in Figure 4. It is a physical system that maintains the attached pole l in a vertical position by moving M in the left and right directions on a frictionless track. In this study, a simulation was performed using the CartPole-v1 environment of the OpenAI Gym toolkit. The state space in question comprised a 4D real number vector, constituting the position and velocity of M and the angle and angular velocity of l. The action space in question comprised a 2D discrete vector constituting the mechanism to push the cart left and right. The end conditions of the environment were as follows: (1) The position of cart M is outside ± 2.4, (2) the angle of pole l is outside ± 24, and (3) the step reward exceeds 500. When an end condition was not met, the agent received a reward of one for each time step. In this environment, the optimal solution for DNNs is to achieve a cumulative reward of 500 according to the maximum time step. Table 1 presents the environment characteristics of the cart-pole balancing experiment. Cart-pole balancing's objective function as follows:

$$\text{Maximize } \sum_{t=1}^{T} Env^R(s_t, \theta(s_t)), \tag{5}$$

where $s_{t+1} = Env^N(s_t, \theta(s_t))$ and $Env^R(s_u, \theta(s_u)) = 0$ for $u \geq t$ if s_t is not feasible. s_t is agent's state vector at time step t. θ is an agent DNN, returning an action vector that maximizes the cumulative rewards when s_t is an input. Env^R and Env^N are simulation functions that receive state and action vectors at each time step. Env^R returns a reward of time step, and Env^N returns a state at next time step. T is a variable representing the maximum time step value 500.

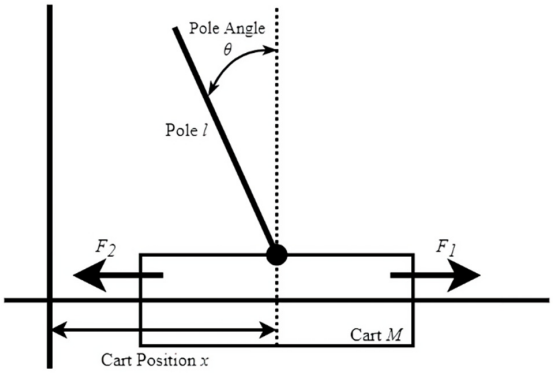


Figure 4. Example of the cart-pole balancing problem environment.

Table 1. Environment of the cart-pole balancing problem.

Environment Parts	Value
Action space	Discrete (2)
Observation shape	Continuous (4)
Continuous observation range	$\pm$[2.4, ∞, 24, ∞]

The following describes the data for surrogate modeling and the design of SGAs in the cart-pole balancing environment in more detail:

1. Surrogate Modeling Data: For designing a surrogate model, pairs of DNN parameters of agents and cumulative reward data according to the DNNs are required. The experimental points were collected through DDQN, which is an off-policy DRL technique. To improve the encoding and computational speed of the SGA, we set it to have a simple DNN architecture. The minimal topology for collecting good solutions was 1 hidden layer and 3 hidden nodes. To measure the performance of the proposed SGAs by network topologies, we set up 4 architecture cases as follows: the number of hidden layers is 1 or 2, and the number of hidden nodes in each layer is 3 or 5. The network collected by DDQN is a fully connected network, and the number of the input and output nodes of cart-pole balancing is 4 and 2, respectively. Therefore, the number of parameters per architecture case are 23 ($= 4 \times 3 + 3 + 3 \times 2 + 2$), 37 ($= 4 \times 5 + 5 + 5 \times 2 + 2$), 35 ($= 4 \times 3 + 3 + 3 \times 3 + 3 + 3 \times 2 + 2$), and 67 ($= 4 \times 5 + 5 + 5 \times 5 + 5 + 5 \times 2 + 2$), respectively. A total of 10,000 data were collected through DDQN learning.
2. Surrogate-assisted GA for the Cart-pole Balancing Problem: Following the stepwise improvements proposed in Section 3.1.4, we designed surrogate models using the surrogate modeling data. In Step 1, two classifiers and one regressor were combined into multiple heuristic surrogates. The classifiers were designed as DNN-based, and each decision boundary was set as 195 and the median value of the experimental points exceeding 195. The first decision boundary of the classifiers, 195, was the threshold of the previous version, CartPole-v0. The regressor of the initial experiment was a DNN that is a nonlinear method commonly used in surrogates. However, in the GA performance to be described later, DNN cannot search the solution that allows it to achieve the maximum reward of 500 for all network architectures. To improve our SGAs, we replaced the regressor with support vector regression (SVR). SVR is a method that applies support vector machine to regression, and it adjusts the regression line to adding as much data as possible inside the margin. In addition, SVR is robust to outliers and overfitting through an ε-incentive loss function. In Step 2, the learning data of the surrogate was adjusted through standardization. In Step 3, the domain of the surrogate was reduced to 2D through PCA, and 10,000 experimental points were added with the same number as the training data.

Table 2 lists the operators and parameters of the SGA used for the cart-pole balancing problem. We used the surrogate models designed through stepwise improvements for fitness evaluation and computed 30 runs in parallel. The fitness and simulation results of 30 runs were confirmed, and their performances were compared according to the network architecture and surrogate model. Our SGAs' experiments were conducted in the environment of Table 3.

Table 2. Operators and parameters of the proposed SGA for the cart-pole balancing problem.

Operator/Parameter	Value
Population size	100
Number of generations	1000
Chromosome lengths	23/37/35/67
Selection method	Roulette-wheel
Crossover method	Extended-box

Table 2. *Cont.*

Operator/Parameter	Value
Mutation rate	1.5%
Replacement	Elitism & Generational

Table 3. Experiments environments.

CPU (core)	AMD Ryzen Threadripper 2990WX (32)
RAM	64 GB
Operating system	Ubuntu 18.04 LTS
Programming language (version)	Python (3.7.11)

3.2.3. Lunar Lander

The lunar lander is a classical rocket trajectory problem, wherein the trajectory of the landing craft must be adjusted to ensure that it lands on the landing pad without collision. As shown in Figure 5, the simulation was conducted using LunarLander-v2, a Gym OpenAI Box2D gaming environment. It is evident that the state and action spaces are more complicated than the cat-pole balancing environment, making the agent's DNN more complex. The state space is an 8D mixed vector comprising the lander coordinates, speed, angle, angular speed, and ground attachment. The action space is a 4D discrete vector comprising do nothing, fire left orientation engine, fire right orientation engine, and fire main engine actions. The agent end condition was configured as follows: (1) the lander touches the ground, (2) the lander is out of the specified x range, and (3) the time step achieves 1000. If the lander reached the landing pad, the agent received a reward; if it moved away, the reward was lost. Additional rewards could be scored based on whether the leg of the lander was in contact with the ground. Finally, if the lander used an engine, the agent lost the reward. The threshold for solving the lunar lander problem was that the average reward of the simulation exceeded 200, which is evaluated as the optimal solution in this environment. The environment characteristics of the lunar lander problem is presented in Table 4. Lunar lander's objective function as follows:

$$\text{Maximize } \sum_{t=1}^{T} Env^R(s_t, \theta(s_t)) + F(s_{T'}), \tag{6}$$

where $s_{t+1} = Env^N(s_t, \theta(s_t))$, $Env^R(s_u, \theta(s_u)) = 0$ for $u \geq t$ if s_t is not feasible, and we meet the end condition, we set T' as t. s_t is agent's state vector at time step t. θ is an agent DNN, returning an action vector that maximizes the cumulative rewards when s_t is an input. The output of Env^R is a loss of the reward calculated through the input of agent's actions (Landers' engine usage). Env^N takes a pair of action and state as input and returns the next state vector at next time step. T is the maximum time step value, and 1000 is used in lunar lander's environment. F receives $s_{T'}$ that satisfies the end condition as an input and returns the agent's reward.

SGA for the Lunar Lander Problem: Similar to the method described in Section 3.2.2, in the lunar lander problem, we attempted to construct the agent's DNN architecture in a simple manner; however, the architecture that satisfied the threshold comprised 2 hidden layers and 32 hidden nodes. The number of parameters of the designed DNN was 1476, which was approximately 22 times higher than that of the structure with the largest number of parameters in the experimental environment of the cart-pole balancing problem. This may degrade computational speed and surrogate performance. Therefore, we measured the Gini importance for all DNN parameters and used the top 100 parameters in the surrogate model design. In the lunar lander experiment, we designed the surrogate model with Step 3 using SVR, which performed best for the cart-pole balancing problem. The same operator and parameters used in the cart-balancing problem were used in the SGAs for the lunar lander problem, and 30 runs were executed in parallel. In the case of the lunar lander problem, even if optimization is performed through the proposed SGAs, it is necessary to

set the remaining 1376 parameters for the simulation. To solve this problem, a group of upper experimental points exceeding the threshold value of 200 was selected. We measured the Euclidean distance of all experimental points of the upper group and the solutions calculated through the SGAs. The simulation was conducted using the parameters of the experimental point that showed the highest similarity.

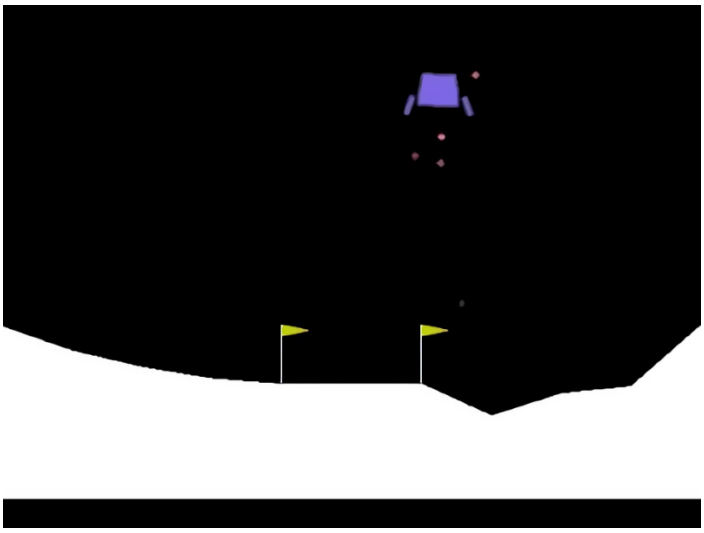


Figure 5. Example of the lunar lander problem environment.

Table 4. Lunar lander problem environment.

Environments Parts	Value
Action space	Discrete (4)
Observation shape	Mixed (Continuous: 6; Bool: 2)
Continuous observation range	±[1.5, 1.5, 5, 5, 3.14, 5]

To help readers reimplement our study, the source codes about our SGAs are available at the site: https://github.com/GoojungMyeon/Mathematics-Surrogate-assisted-Genetic-algorithm (accessed on 20 November 2022).

4. Results and Discussion

4.1. Surrogate Modeling Data

The cumulative reward distribution for each architecture case is shown in Figure 6. The experimental points that achieved the maximum cumulative reward of 500 were collected for all architecture cases. In the simplest case (1, 3), the solution quality was worst, and the most returns were lower than the decision threshold of 195 for the first classifier of the surrogate model described in the cart-pole balancing problem.

4.2. Surrogate-Assisted GA for the Cart-Pole Balancing Problem

We trained and evaluated collecting data of the cart-pole balancing according to stepwise improvements as shown in Table A2 (see Appendix B). In Step 2, wherein the learning data of the surrogate was adjusted through standardization, improvements in both the root mean square error (RMSE) and correlation coefficient were confirmed, except in the (2, 3, 3) case. For all architectures, the RMSE values were improved compared with Step 2, and for the (1, 5) case, the RMSE value in Steps 1–3 was improved the most to 19.2%. Figure 7 shows the RMSE and correlation coefficient values of surrogate models according to the architectures and stepwise improvements.

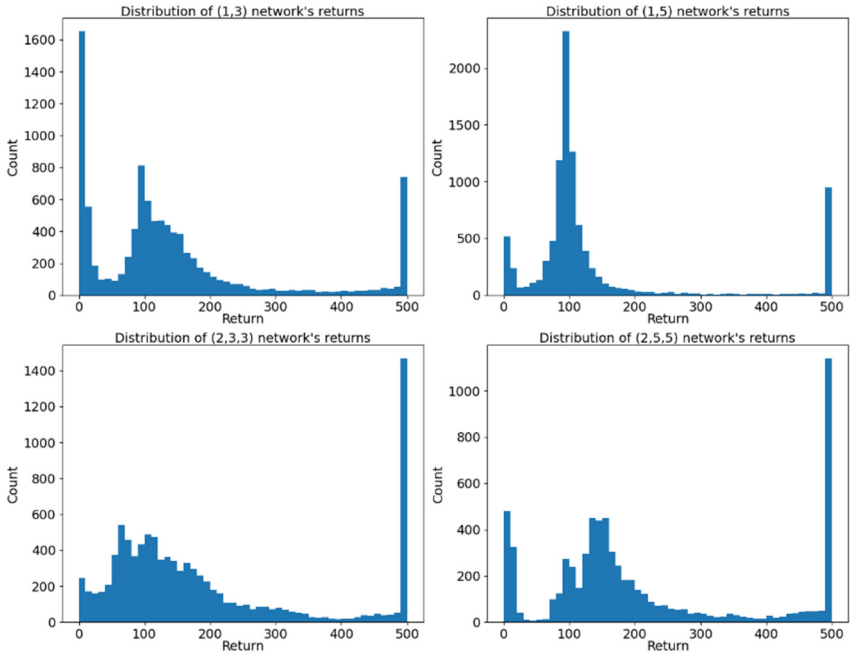
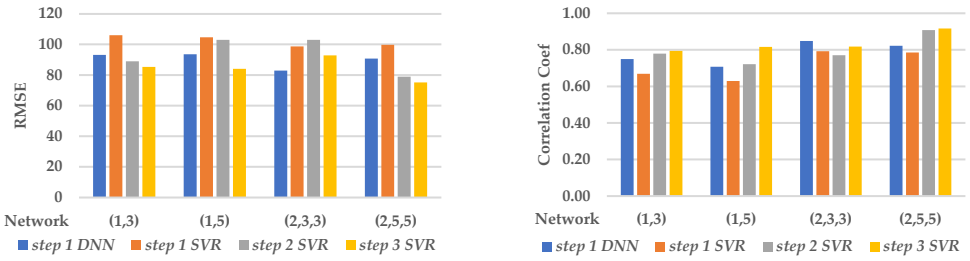


Figure 6. Return distribution of each neural network architecture.



(a) (b)

Figure 7. Performance for each architecture with stepwise improvement. (**a**) RMSE and (**b**) Pearson correlation coefficient.

Table 5 lists the performance results of the proposed SGA for the cart-pole balancing problem according to the surrogate model and the network. It presents the fitness and simulation results of 30 runs. In addition, the hypothesis test results between simulation values according to the SGAs' surrogate model are also shown. Since the simulation values of SGA are nonparametric and independent, the method of a hypothesis test is Mann–Whitney U; most fitness scores are overestimated compared to the simulation score. In Step 1, multiple surrogates were used, and the regressor was configured as DNN or SVR. When the DNN regressor was used, the maximum cumulative reward could not be determined in all the architecture cases. However, the agent DNN achieved a maximum cumulative reward of 500 in all cases except (1, 3) when the SVR regressor was used. This suggests that in the data distribution discussed in Section 4.1, the (1, 3) case achieved a low optimization performance because many solutions did not satisfy the decision boundary of the first classifier. The results obtained using the Step 2 surrogate model that applied input data standardization were enhanced for all architectures. In the (1, 5) network, the simulation scores of all runs were 500. The last step improved the results for all networks, and an optimum value of 500 was obtained by all runs in the (1, 5) network. The p-value is a significant probability, and it can be analyzed that the difference in variance between two samples is significant when p-value is less than 0.05. When the regressor was changed from DNN to SVR and improvement on Step 2 was applied, p-values was less than 0.5 except the (1, 3) case. Figure 8 shows the computing times of the proposed SGAs and the

baseline DDQN for cart-pole balancing optimization. Through DDQN, which was used for collecting experimental points, the time required for searching the maximum cumulative reward was set to the baseline. Although the computing time of the SGAs increased when the stepwise improvement was applied, it is shown that the computing time was reduced in all architectures rather than in baseline. The computing time was reduced in all network experiments using the proposed SGAs, and in the best optimization (1, 5) it was reduced by approximately 38%.

Table 5. Results of the SGAs for the cart-pole balancing problem according to the network architecture and surrogate model (from 30 runs).

Surrogate Model	Network	Fitness			Simulation			
		Best	Average	STD	Best	Average	STD	*p*-Value
DNN *step* 1	(1, 3)	762.602	748.419	77.334	9.480	9.359	0.070	-
	(1, 5)	2636.006	2414.450	412.272	88.470	11.990	14.202	-
	(2, 3, 3)	1936.146	1893.751	32.744	27.290	13.729	5.578	-
	(2, 5, 5)	13,117.208	12,286.410	295.666	9.480	9.345	0.080	-
SVR *step* 1	(1, 3)	682.976	682.510	0.276	9.570	9.357	0.075	7.56×10^{-1}
	(1, 5)	991.123	920.751	151.260	500.000	333.722	184.831	2.46×10^{-11}
	(2, 3, 3)	969.271	942.010	92.769	500.000	42.093	122.381	4.62×10^{-3}
	(2, 5, 5)	637.732	614.532	12.041	500.000	165.670	167.801	7.66×10^{-10}
SVR *step* 2	(1, 3)	649.203	649.139	0.044	13.820	11.937	0.773	2.98×10^{-11}
	(1, 5)	585.734	573.516	3.355	500.000	500.000	0.000	5.37×10^{-6}
	(2, 3, 3)	574.514	557.821	47.189	500.000	469.507	73.103	5.29×10^{-9}
	(2, 5, 5)	534.788	514.015	6.391	500.000	327.571	199.072	7.00×10^{-4}
SVR *step* 3	(1, 3)	649.197	649.145	0.036	25.850	13.065	2.677	9.26×10^{-3}
	(1, 5)	578.489	574.012	2.166	500.000	500.000	0.000	N.A.
	(2, 3, 3)	574.635	567.215	17.360	500.000	491.200	20.261	1.22×10^{-1}
	(2, 5, 5)	534.798	510.536	6.585	500.000	259.955	201.400	1.80×10^{-1}

Note: The results of "***Fitness***" were calculated by each SGA's surrogate model, and the results of "***Simulation***" were calculated by a real simulator of cart-pole balancing.

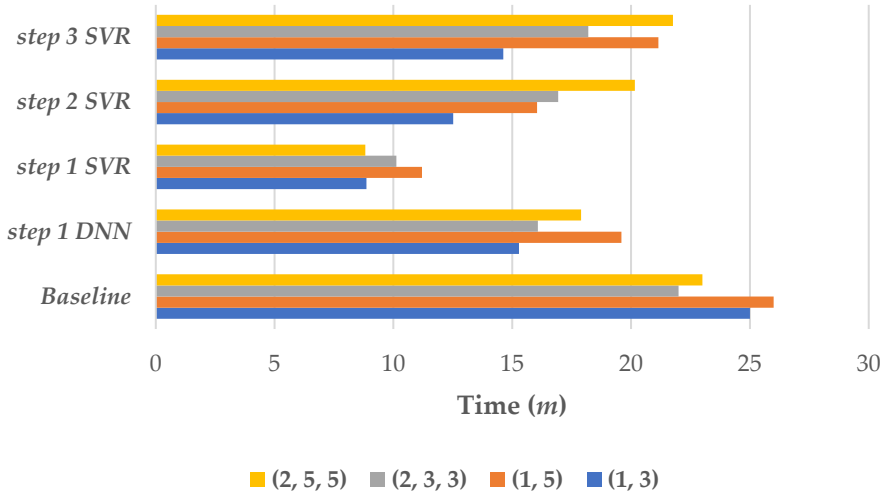


Figure 8. Calculation times of the baseline DDQN and the proposed SGAs for cart-pole balancing optimization.

4.3. Results for the Lunar Lander Problem

Like the environment of the cart-pole balancing, our SGAs in the Lunar lander also reduced computing time compared to the baseline. The baseline is the initial discovery time of the achievement of a problem via DDQN for collecting experimental points. The average computing time of 30 runs of our SGA was approximately 18 min and the calculation time of the baseline was 363 min, which was improved approximately 95% over the baseline.

Among the 30 runs, 24 exceeded the problem-solving threshold, and the best-performing SGAs explored agent DNNs, which were approximately 4% better than the best solution of the training data of the surrogate (Table 6). Figure 9 shows the best individual average for each generation of the 30 runs.

Table 6. Simulation performance of the baseline and the proposed SGAs for the lunar lander problem (from 30 runs).

		Best Reward	Mean Reward
Baseline		303.452	216.66 ± 5.056
Proposed SGAs	Best	308.930	224.665 ± 5.657
	Mean	290.352	155.297 ± 8.047

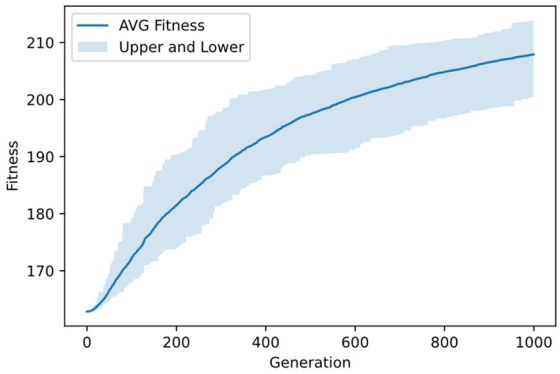


Figure 9. Fitness of the proposed SGAs for the lunar lander problem according to generation.

5. Conclusions

In this study, we proposed novel SGAs for agent's DNN parameter optimization and conducted experiments to determine an optimal agent. The goal was to reduce computing time over simulation-based methods and to identify an optimal agent DNN. We designed a surrogate model to predict cumulative rewards for agent's DNN parameters, wherein we proposed three stepwise improvements to prevent convergence misleading, which are disadvantages of existing surrogate-assisted methods. The first step involved using multiple surrogates to prevent the convergence of the approximate performance and false optimum. In the second step, data standardization was applied to prevent performance degradation owing to outliers in the experimental points. Finally, random sampling with PCA was applied to prevent performance degradation owing to the high-dimensional nature and limited number of experimental points. The improvement in performance owing to these stepwise improvements was confirmed through experiments in a cart-pole balancing environment. We identified agents that achieved the maximum reward in all SGAs except for the surrogate model that did not have good-quality experimental points, and reduced the computing time by 38% compared to the DDQN. We also proposed using SGAs in complex problem environments, such as the lunar lander. We proposed a strategy to measure the importance of each parameter when the number of agent's DNN parameters is high and to design a surrogate model with only some parameters. Consequently, we designed a surrogate model with only 100 parameters selected by Gini importance among the existing 1476, and our SGAs' performances were better than that of the baseline. The computing time was reduced by approximately 95%, and the quality was even improved by approximately 4%. Through the above experiments, we confirmed that our SGAs can be applied to any discrete and deterministic environment of the action space such as *mountain car* or *Atari video games*. However, since our SGAs are offline methods in which high-quality experimental points must be preceded, their performance may be poor if the efficiency of collecting experimental points is low.

In future research, three types of expansion are possible: more complex problems, online learning, and genetic process control. In this study, the action space employed was discrete and deterministic. However, in future studies, it could be applied to a continuous or stochastic action space for experimentation. Additionally, we will investigate whether or not it is applicable to partially observed environments. In this study, the offline method was adopted to collect experimental points; however, this method was inefficient when the quality of the experimental points was poor or the collection cost was high. Therefore, in future experiments, the online learning method of updating the surrogate-assisted for optimization during genetic process will be applied. Finally, the performance improvement will be confirmed through intelligent evolution control, which evaluate specific chromosomes with a real fitness function in the population.

Author Contributions: Conceptualization, Y.-H.K.; methodology, Y.-H.K.; software, S.-S.S.; validation, S.-S.S.; formal analysis, Y.-H.K.; investigation, S.-S.S.; resources, Y.-H.K.; data curation, Y.-H.K.; Writing—Original draft preparation, S.-S.S.; Writing—Review and editing, Y.-H.K.; visualization, S.-S.S.; supervision, Y.-H.K.; project administration, Y.-H.K.; funding acquisition, Y.-H.K. All authors have read and agreed to the published version of the manuscript.

Funding: The present research has been conducted by the Research Grant of Kwangwoon University in 2022. This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1048466).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Experiments for Surrogate-Assisted Genetic Algorithms According to the Number of Surrogates Samples in the Cart-Pole Balancing

In this appendix, experiments were conducted to examine the appropriate number of samples for the surrogates' design in cart-pole balancing environment. When each surrogate is applied to a GA, the performance is compared. Surrogates in these experiments were designed with Step 3 SVR in Section 3.1.4. The SGAs for this pilot test used the operators and parameters in Section 3 and performed 30 independent runs. The performances of SGAs according to the number of samples are shown in Table A1. In all the cases, the optimal solution with real fitness 500 was found. In addition, it is shown that the average values among 30 runs of SGA increased as increasing of the sample's number. Therefore, we set the number of samples in our SGAs to 10,000.

Table A1. Performance of SGAs based on the number of samples in the cart-pole balancing.

The Number of Samples	Fitness			Simulation		
	Best	Average	STD.	Best	Average	STD.
1000	554.068	551.114	2.077	500.000	437.360	138.333
3000	611.015	605.590	3.745	500.000	467.872	95.159
6000	548.681	543.419	2.957	500.000	469.741	82.721
10,000	578.489	574.012	2.166	500.000	500.000	0.000

Appendix B. Results of the Proposed Surrogates on Each Network Architecture with Stepwise Improvement in the Cart-Pole Balancing

In this appendix, we provide performances of our surrogates on each network architecture with stepwise improvement in the cart-pole balancing. Table A2 shows root mean square error (RMSE), mean percentage error (MPE), mean absolute percentage error (MAPE), and Pearson coefficient (Coef.) values according to surrogate and network architecture.

Table A2. Results of surrogate models in the cart-pole balancing.

***Step* 1-DNN**																
Network	(1, 3)/23				(1, 5)/37				(2, 3, 3)/35				(2, 5, 5,)/67			
Measurement	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.
Test	93.17	−101.45	120.85	0.75	93.52	−52.90	70.39	0.71	82.89	−41.53	60.48	0.85	90.79	−29.88	46.44	0.82
Train	90.02	−86.51	106.05	0.77	82.83	−45.58	61.20	0.78	76.57	−39.54	56.16	0.87	71.74	−22.42	35.92	0.90
***Step* 1-SVR**																
Network	(1, 3)/23				(1, 5)/37				(2, 3, 3)/35				(2, 5, 5,)/67			
Measurement	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.
Test	105.99	−95.89	120.37	0.67	104.70	−42.78	64.27	0.63	98.72	−51.71	69.44	0.79	99.72	−41.02	54.95	0.79
Train	105.29	−80.65	106.10	0.68	100.95	−37.04	57.84	0.66	98.09	−49.99	67.32	0.80	92.54	−38.60	51.81	0.82
***Step* 2-SVR**																
Network	(1, 3)/23				(1, 5)/37				(2, 3, 3)/35				(2, 5, 5,)/67			
Measurement	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.
Test	88.92	−72.31	95.11	0.78	103.02	−52.18	80.33	0.72	103.00	−34.63	59.08	0.77	78.88	−46.47	66.19	0.91
Train	83.70	−65.83	89.29	0.79	102.12	−48.24	77.56	0.75	102.55	−34.22	57.86	0.78	77.48	−45.11	64.04	0.92
***Step* 3-SVR**																
Network	(1, 3)/23				(1, 5)/37				(2, 3, 3)/35				(2, 5, 5,)/67			
Measurement	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.	RMSE	MPE	MAPE	Coef.
Test	85.31	−65.79	87.02	0.79	84.04	−53.74	80.78	0.82	92.84	−27.47	47.75	0.82	75.16	−42.36	60.92	0.92
Train	80.73	−59.10	81.09	0.79	82.43	−50.60	78.17	0.81	92.40	−27.49	46.78	0.82	72.26	−41.16	58.71	0.92

References

1. Stuart, J.R.; Peter, N. *Artificial Intelligence: A Modern Approach*, 4th ed.; Prentice Hall: Hoboken, NJ, USA, 2020.
2. Otterlo, M.V.; Wiering, M. Reinforcement learning and Markov decision processes. In *Reinforcement Learning*; Wiering, M., van Otterlo, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 12, pp. 3–42. [CrossRef]
3. Such, F.P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K.O.; Clune, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv* **2017**, arXiv:1712.06567.
4. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–9126. [CrossRef] [PubMed]
5. Mirjalili, S. Genetic Algorithm. In *Evolutionary Algorithms and Neural Networks*; Springer: Cham, Germany, 2019; pp. 43–55. [CrossRef]
6. Jin, Y. Surrogate-assisted evolutionary computation: Recent advanced and future challenges. *Swarm Evol. Comput.* **2011**, *1*, 61–70. [CrossRef]
7. Pei, Y.; Gao, H.; Han, X. A Surrogate Model Based Genetic Algorithm for Complex Problem Solving. In Proceedings of the 6th Annual International Conference on Network and Information Systems for Computers (ICNISC2020), Guiyang, China, 14–15 August 2020.
8. Cho, D.H.; Moon, S.H.; Kim, Y.H. Genetic feature selection applied to KOSPI and cryptocurrency price prediction. *Mathematics* **2021**, *9*, 2574. [CrossRef]
9. Kim, Y.H.; Yoon, Y.; Kim, Y.H. Towards a better basis search through a surrogate model-based epistasis minimization for pseudo-Boolean optimization. *Mathematics* **2020**, *8*, 1287. [CrossRef]
10. Cho, H.Y.; Kim, Y.H. A Genetic Algorithm to Optimize SMOTE and GAN Ratios in Class Imbalanced Datasets. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Cancun, Mexico, 8–12 July 2020; pp. 33–34. [CrossRef]
11. Cai, X.; Gao, L.; Li, X. Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems. *IEEE Trans. Evol. Comput.* **2020**, *24*, 365–379. [CrossRef]
12. Francon, O.; Gonzalez, S.; Hodjat, B.; Meyerson, E.; Miikkulainen, R.; Qiu, X.; Shahrzad, H. Effective Reinforcement Learning through Evolutionary Surrogate-Assisted Prescription. In Proceedings of the Genetic and Evolutionary Computation Conference, Cancun, Mexico, 8–12 July 2020; pp. 814–822. [CrossRef]
13. Yu, D.P.; Kim, Y.-H. Predictability on Performance of Surrogate-Assisted Evolutionary Algorithm According to Problem Dimension. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 13–17 July 2019; pp. 91–92. [CrossRef]
14. Jin, Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.* **2005**, *9*, 3–12. [CrossRef]
15. Calisto, M.B.; Lai-Yuen, S.K. EMONAS-Net: Efficient multiobjective neural architecture search using surrogate-assisted evolutionary algorithm for 3D medical image segmentation. *Artif. Intell. Med.* **2019**, *119*, 102154. [CrossRef] [PubMed]
16. Sun, Y.; Wang, H.; Xue, B.; Jin, Y.; Ten, G.G.; Zhang, M. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Tran. Evol. Comput.* **2019**, *24*, 350–364. [CrossRef]
17. Pholdee, N.; Baek, H.M.; Bureerat, S.; Im, Y.T. Process optimization of a non-circular drawing sequence based on multi-surrogate assisted meta-heuristic algorithms. *J. Mech. Sci. Technol.* **2015**, *29*, 3427–3436. [CrossRef]

18. Vincenzi, L.; Gambarelli, P. A proper infill sampling strategy for improving the speed performance of a surrogate-assisted evolutionary algorithm. *Comput. Struct.* **2017**, *178*, 58–70. [CrossRef]
19. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998. [CrossRef]
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
21. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with A Stochastic Actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
22. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
23. Hausknecht, M.; Lehman, J.; Miikkulainen, R.; Stone, P. A neuroevolution approach to general Atari game playing. *IEEE Tran. Comput. Intell. AI* **2014**, *6*, 355–366. [CrossRef]
24. Stanley, K.O.; Clune, J.; Lehman, H.; Miikkulainen, R. Designing neural networks through neuroevoultion. *Nat. Mach. Intell.* **2019**, *1*, 24–35. [CrossRef]
25. Van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double *Q*-Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 14–17 February 2016; pp. 2094–2100. [CrossRef]
26. Clune, J.; Stanley, K.O.; Pennock, R.T.; Ofria, C. On the performance of indirect encoding across the continuum of regularity. *IEEE Trans. Evol. Comput.* **2011**, *15*, 346–367. [CrossRef]
27. Yoon, Y.; Kim, Y.H.; Moraglio, A.; Moon, B.R. A theoretical and empirical study on unbiased boundary-extended crossover for real-valued representation. *Inf. Sci.* **2012**, *183*, 48–65. [CrossRef]
28. Shin, S.S.; Kim, Y.H. A surrogate model-based genetic algorithm for the optimal policy in cart-pole balancing environments. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Boston, MA, USA, 9–13 July 2022; pp. 503–505. [CrossRef]
29. Zheng, N.; Wang, H.; Yuan, B. An adaptive model switch-based surrogate-assisted evolutionary algorithm for noisy expensive multi-objective optimization. *Complex Intell. Syst.* **2022**, *8*, 4339–4356. [CrossRef]
30. Shaukat, S.S.; Rao, T.A.; Khan, M.A. Impact of sample size on principal component analysis ordination of an environmental data set: Effects on eigenstructure. *Ekológia* **2016**, *35*, 173. [CrossRef]