Genetic Algorithm with Machine Learning to Estimate the Optimal Objective Function Values of Subproblems

Hitoshi Iima
Kyoto Institute of Technology, Matsugasaki, Sakyo-ku, Kyoto, Japan
iima@kit.ac.jp

Yohei Hazama
Kyoto Institute of Technology, Matsugasaki, Sakyo-ku, Kyoto, Japan
hazama0y@cis.is.kit.ac.jp

ABSTRACT

This paper addresses an optimization problem with two decision variable vectors. This problem can be divided into multiple subproblems when an arbitrary value is given to the first decision variable vector. In conventional genetic algorithms (GAs) for the problem, an individual is often expressed by the value of the first decision variable vector. In evaluating the individual, the value of the remaining decision variable vector is determined by metaheuristics or greedy algorithms. However, such GAs are time-consuming or not general-purpose. We propose a GA with a neural network model to estimate the optimal objective function values of the subproblems. Experimental results compared to other GAs show that the proposed method is effective.

CCS CONCEPTS

• **Computing methodologies** → Machine learning; Machine learning approaches; Bio-inspired approaches; Genetic algorithms; Machine learning; Machine learning approaches; Neural networks.

KEYWORDS

Genetic algorithm, Neural network, Optimization, Machine learning, Surrogate model

ACM Reference Format:
Hitoshi Iima and Yohei Hazama. 2022. Genetic Algorithm with Machine Learning to Estimate the Optimal Objective Function Values of Subproblems. In *2022 6th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI 2022), April 09, 10, 2022, Seoul, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3533050.3533051

1 INTRODUCTION

The genetic algorithm (GA) [1] is known as an effective method for solving optimization problems. It is one of the metaheuristics using multiple candidate solutions to find a near-optimal solution in a reasonable time and has been actively studied. In recent years, more complicated optimization problems have needed to be solved, and it is important to develop GAs that can find a better solution in a shorter time.

ISMSI 2022, April 09, 10, 2022, Seoul, Republic of Korea

ACM ISBN 978-1-4503-9628-8/22/04…$15.00
https://doi.org/10.1145/3533050.3533051

When a GA is applied to an optimization problem, especially to one with two decision variable vectors, the following strategy is often adopted [2–7]. An individual of GA is expressed by the values of one of the decision variable vectors. In evaluating the individual, the values of the remaining decision variable vector are determined by applying metaheuristics or greedy algorithms. This application corresponds to solving the problem of optimizing the remaining decision variable vector. However, the metaheuristics are time-consuming, and the greedy algorithms are not general-purpose.

In this paper, we address a problem in which optimizing the remaining decision variable vector can be divided into multiple similar subproblems, and we propose a GA with a neural network for solving the problem. As mentioned above, in the individual evaluation of existing GAs, metaheuristics or greedy algorithms intend to solve the subproblems: find the optimal values of their decision variables. However, what is needed in the individual evaluation is the optimal values of objective functions, not decision variables. Therefore, in the proposed method, the neural network estimates the optimal objective function values of the subproblems, and the estimated values are used for evaluating the individual. This estimation method is general-purpose and is not time-consuming.

As another GA with machine learning, the surrogate-assisted GA is proposed [8–10]. This GA uses the surrogate model to approximate an objective function, and the surrogate model is trained during the execution of the GA. In contrast, the proposed method uses a model to estimate the optimal objective function values of subproblems, and the model is trained before the execution of the GA.

The rest of this paper is organized as follows. Section 2 defines a problem to be solved in this paper and shows its concrete examples. Section 3 proposes a method for solving the problem. Section 4 shows experimental results for one of the concrete examples and evaluates the performance of the proposed method by comparison with other methods. Finally, Section 5 concludes this paper.

2 PROBLEM STATEMENT

This section defines a problem to be solved in this paper and then shows its concrete examples.

2.1 Definition

We address a problem with two decision variable vectors, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The problem is formulated as

$$\min_{\boldsymbol{x}_1, \boldsymbol{x}_2} f(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{P}), \tag{1}$$

where $\boldsymbol{P}$ is a parameter vector in the problem. We give two examples of $\boldsymbol{P}$: in the knapsack problem, $\boldsymbol{P}$ consists of the values of

items, and in the traveling salesman problem, $\boldsymbol{P}$ consists of distances between cities. f is the objective function and is parametrized by $\boldsymbol{P}$.

Once the first decision variable vector $\boldsymbol{x}_1$ is given an arbitrary value $\boldsymbol{a}_1$, f can be expressed as the sum of partial functions g^k $(k = 1, 2, \ldots, q)$ and is given by

$$f(\boldsymbol{a}_1, \boldsymbol{x}_2; \boldsymbol{P}) = \sum_{k=1}^{q} g^k\left(\boldsymbol{x}_{2\boldsymbol{k}}; \boldsymbol{P}_{\boldsymbol{k}}\right), \tag{2}$$

where each $\boldsymbol{x}_{2\boldsymbol{k}}$ is a part of the second decision variable vector $\boldsymbol{x}_2$. Each element of $\boldsymbol{x}_2$ must be an element of one of $\boldsymbol{x}_{21}, \boldsymbol{x}_{22}, \ldots, \boldsymbol{x}_{2\boldsymbol{q}}$, and the correspondence depends on $\boldsymbol{a}_1$. $\boldsymbol{P}_{\boldsymbol{k}}$ is a part of $\boldsymbol{P}$, and the correspondence also depends on $\boldsymbol{a}_1$. The partial function g^k is a function of $\boldsymbol{x}_{2\boldsymbol{k}}$ and is parametrized by $\boldsymbol{P}_{\boldsymbol{k}}$. Since all g^k are independent, minimizing f in equation (2) can be achieved by minimizing each g^k individually. The subproblem of minimizing g^k is formulated as

$$\min_{\boldsymbol{x}_{2\boldsymbol{k}}} g^k\left(\boldsymbol{x}_{2\boldsymbol{k}}; \boldsymbol{P}_{\boldsymbol{k}}\right) \ (k = 1, 2, \ldots, q). \tag{3}$$

All subproblems are assumed to be similar: all $\boldsymbol{P}_{\boldsymbol{k}}$ consist of the same kind of parameters as each other, and all g^k are of the same form.

2.2 Concrete Examples

Concrete examples of the problem defined in Subsection 2.1 include the clustering problems [3, 5] and the drone delivery scheduling problem (drone problem) [11]. This subsection describes one of the clustering problems and the drone problem.

2.2.1 Clustering problem [5]. In a service area, n users are divided into q clusters to allocate power to them. They receive the power from a base station. Users in each cluster share the power resources of the cluster. The clustering problem is to determine the cluster to which each user belongs and the amount of power allocated to the user to maximize the total throughput from the base station to all users.

Let y_i be the decision variable meaning the cluster number to which the user i $(= 1, 2, \ldots, n)$ belongs, z_i be the decision variable meaning the amount of power allocated to the user i, and h_i be the channel gain to the user i. The decision variable vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and the parameter vector $\boldsymbol{P}$ defined in Subsection 2.1 are as follows:

$$\boldsymbol{x}_1 = [y_1\ y_2 \ldots\ y_n],$$

$$\boldsymbol{x}_2 = [z_1\ z_2 \ldots\ z_n],$$

$$\boldsymbol{P} = [h_1\ h_2 \ldots\ h_n].$$

Once $\boldsymbol{x}_1$ is given an arbitrary value $\boldsymbol{a}_1$, the objective function f can be divided into q partial objective functions each of which is the sum of throughputs to users in one cluster. Specifically, let $n(k)$ be the number of users in the cluster k $(= 1, 2, \ldots, q)$, that is, the number of variables y_i satisfying $y_i = k$. Note that these numbers $n(k)$ depend on $\boldsymbol{a}_1$. Let $i_m^k \in \{1, 2, \ldots, n\}$ $(m = 1, 2, \ldots, n(k))$ be the m-th user in the cluster k. The partial objective function g^k, the decision variable vector $\boldsymbol{x}_{2\boldsymbol{k}}$, and the parameter vector $\boldsymbol{P}_{\boldsymbol{k}}$ are as follows:

$$g^k\left(\boldsymbol{x}_{2\boldsymbol{k}}; \boldsymbol{P}_{\boldsymbol{k}}\right) = -\sum_{m=1}^{n(k)} N_1 \log_2\left(1 + \frac{z_{i_m^k} h_{i_m^k}}{\sum_{m'=1}^{n(k)} z_{i_{m'}^k} h_{i_{m'}^k} + N_2}\right),$$

$$\boldsymbol{x}_{2\boldsymbol{k}} = \left[z_{i_1^k} \ \ldots z_{i_{n(k)}^k}\right],$$

$$\boldsymbol{P}_{\boldsymbol{k}} = \left[h_{i_1^k} \ldots\ h_{i_{n(k)}^k}\right],$$

where N_1 and N_2 are constants. Each subproblem is to determine the amount of power allocated to each user in one cluster. The power allocation in each cluster does not affect the other clusters, and therefore each subproblem can be optimized individually. For example, by letting $n = 6, q = 3$, and $\boldsymbol{x}_1 = [1\ 2\ 3\ 1\ 3\ 2]$, $i_1^1 = 1, i_2^1 = 4, i_1^2 = 2, i_2^2 = 6, i_1^3 = 3$, and $i_2^3 = 5$ are determined. Therefore, $\boldsymbol{x}_{21} = [z_1\ z_4]$, $\boldsymbol{x}_{22} = [z_2\ z_6], \boldsymbol{x}_{23} = [z_3\ z_5], \boldsymbol{P}_1 = [h_1\ h_4], \boldsymbol{P}_2 = [h_2\ h_6]$, and $\boldsymbol{P}_3 = [h_3\ h_5]$ are also determined. All $\boldsymbol{P}_{\boldsymbol{k}}$ consist of h_i, and all g^k are of the same form. Therefore, all subproblems are similar.

2.2.2 Drone problem [11]. A truck loaded with n parcels and p drones leaves a distribution center and stops at q points on the way to a destination. The drones take off from each point and deliver parcels to customers. A drone can carry only one parcel. Therefore, if the drones deliver to more than p customers from the point, some of them must return to the truck and deliver undelivered parcels. When all drones finish delivering from the point and return to the truck, the truck moves to the next point. From the last point, the truck moves to the destination.

The drone problem is to determine the drone delivering a parcel to each customer and the point at which the drone takes off. Its objective is to minimize the completion time of delivering to all customers. The route of the truck is fixed, so its travel time does not change. Therefore, minimizing the delivery completion time is the same as minimizing the sum of the longest flight time of all drones at each point.

Let y_i be a decision variable meaning the point number from which a drone delivers to the customer i $(= 1, 2, \ldots, n)$ and z_i be a decision variable meaning the drone number delivering to the customer i. Let w_{ik} be the drone flight time between the customer i and the point k $(= 1, 2, \ldots, q)$. The decision variable vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and the parameter vector $\boldsymbol{P}$ defined in Subsection 2.1 are as follows:

$$\boldsymbol{x}_1 = [y_1\ y_2 \ldots y_n],$$

$$\boldsymbol{x}_2 = [z_1\ z_2 \ldots z_n],$$

$$\boldsymbol{P} = \left[w_{11}\ w_{12} \ldots w_{1q}\ w_{21} \ldots w_{2q} \ldots w_{nq}\right].$$

Once $\boldsymbol{x}_1$ is given an arbitrary value $\boldsymbol{a}_1$, the objective function f can be divided into q partial objective functions each of which is the delivery completion time at one point. Specifically, let $n(k)$ be the number of customers to who drones deliver from the point k, that is, the number of variables y_i satisfying $y_i = k$. Note that these numbers $n(k)$ depend on $\boldsymbol{a}_1$. Let $i_m^k \in \{1, 2, \ldots, n\}$ $(m = 1, 2, \ldots, n(k))$ be the m-th customer to who a drone delivers from the point k. The partial objective function g^k, the decision variable vector $\boldsymbol{x}_{2\boldsymbol{k}}$, and the parameter vector $\boldsymbol{P}_{\boldsymbol{k}}$ are as follows:

$$g^k\left(\boldsymbol{x}_{2\boldsymbol{k}}; \boldsymbol{P}_{\boldsymbol{k}}\right) = \max_{j=1,2,\ldots,p} \sum_{m \in M(j,k)} w_{i_m^k k},$$

$$M(j,k) = \left\{ m \;\middle|\; j = z_{i_m^k} \right\},$$

$$\boldsymbol{x}_{2\boldsymbol{k}} = \left[z_{i_1^k} \;\ldots z_{i_{n(k)}^k} \right],$$

$$\boldsymbol{P}_{\boldsymbol{k}} = \left[w_{i_1^k k} \ldots w_{i_{n(k)}^k k} \right],$$

where $M(j,k)$ is the set of customers to who the drone j delivers from the point k. Each subproblem is to determine the drones that deliver to customers from one point. The flight time from each point does not affect the other points, and therefore each subproblem can be optimized individually. For example, by letting $n = 10, q = 3$, and $\boldsymbol{x}_1 = [1\ 1\ 2\ 3\ 2\ 1\ 2\ 3\ 2\ 1]$, $i_1^1 = 1, i_2^1 = 2, i_3^1 = 6, i_4^1 = 10, i_1^2 = 3, i_2^2 = 5, i_3^2 = 7, i_4^2 = 9, i_1^3 = 4$, and $i_2^3 = 8$ are determined. Therefore, $\boldsymbol{x}_{21} = [z_1\ z_2\ z_6\ z_{10}]$, $\boldsymbol{x}_{22} = [z_3\ z_5\ z_7\ z_9]$, $\boldsymbol{x}_{23} = [z_4\ z_8]$, $\boldsymbol{P}_1 = [w_{11}\ w_{21}\ w_{61}\ w_{10,1}]$, $\boldsymbol{P}_2 = [w_{32}\ w_{52}\ w_{72}\ w_{92}]$, and $\boldsymbol{P}_3 = [w_{43}\ w_{83}]$ are also determined. All $\boldsymbol{P}_{\boldsymbol{k}}$ consist of w_{ik}, and all g^k are of the same form. Therefore, all subproblems are similar.

3 PROPOSED METHOD

This section proposes a GA with a neural network model to estimate $\min g^k$, the optimal objective function value of subproblem (3). We describe the basic concept of the proposed method and then its details.

3.1 Basic Concept

In existing GAs for the problem (1) with two decision variable vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, each individual often consists of the value of only $\boldsymbol{x}_1$. The value of the remaining $\boldsymbol{x}_2$ is determined whenever the individual is evaluated to select good ones used in operations such as the crossover. It is evaluated to be good if the determination of $\boldsymbol{x}_2$ results in generating a good solution. For generating the good solution, the value of $\boldsymbol{x}_2$ is determined by applying metaheuristics or greedy algorithms [2–7]. However, the metaheuristics are time-consuming, and the greedy algorithms are not general-purpose. In addition, although these methods determine the value of $\boldsymbol{x}_2$, rather than the optimal value of $\boldsymbol{x}_2$, the individual evaluation requires $\min_{\boldsymbol{x}_2} f(\boldsymbol{a}_1, \boldsymbol{x}_2)$, the optimal objective function value under $\boldsymbol{x}_1 = \boldsymbol{a}_1$.

In recent years, neural networks have attracted attention again and been actively studied. If a neural network model can estimate $\min_{\boldsymbol{x}_2} f(\boldsymbol{a}_1, \boldsymbol{x}_2)$ from $\boldsymbol{a}_1$, the estimated value can be used to evaluate an individual of GA. The estimation by this model takes a reasonable time and is applicable for all problems formulated as (1), which overcomes the disadvantages of the metaheuristics and the greedy algorithms. However, the model must be trained during the execution of the GA, so the training is time-consuming and is required for each instance. If the model can estimate $\min_{\boldsymbol{x}_2} f(\boldsymbol{a}_1, \boldsymbol{x}_2; \boldsymbol{P})$ from not only $\boldsymbol{a}_1$ but also $\boldsymbol{P}$, it can be used commonly for all instances. Moreover, it can be trained before the execution of the GA, so the execution time does not increase. However, since the number of inputs $\boldsymbol{a}_1$ and $\boldsymbol{P}$ to the model is larger, training the model is more difficult.

As shown in equation (2), once the decision variable vector $\boldsymbol{x}_1$ is given an arbitrary value $\boldsymbol{a}_1$, the objective function f can be expressed as the sum of multiple partial functions $g^k(\boldsymbol{x}_{2\boldsymbol{k}}; \boldsymbol{P}_{\boldsymbol{k}})$. Therefore, instead of directly estimating $\min_{\boldsymbol{x}_2} f(\boldsymbol{a}_1, \boldsymbol{x}_2; \boldsymbol{P})$, our proposed model indirectly estimates $\min_{\boldsymbol{x}_2} f(\boldsymbol{a}_1, \boldsymbol{x}_2; \boldsymbol{P})$ by estimating $g_{min}^k = \min_{\boldsymbol{x}_{2\boldsymbol{k}}} g^k(\boldsymbol{x}_{2\boldsymbol{k}}; \boldsymbol{P}_{\boldsymbol{k}})$ and summing them. Since the parameter $\boldsymbol{P}_{\boldsymbol{k}}$ in the subproblem (3) is determined depending on $\boldsymbol{a}_1$, the proposed model can estimate g_{min}^k from only $\boldsymbol{P}_{\boldsymbol{k}}$ without $\boldsymbol{a}_1$. Moreover, the dimension of $\boldsymbol{P}_{\boldsymbol{k}}$ is lower than that of $\boldsymbol{P}$. Therefore, training the proposed model becomes easier. Since all subproblems are similar as assumed in Subsection 2.1, the proposed model can alone estimate g_{min}^k for all k, and multiple models are not required. Therefore, it is more efficient.

3.2 Basic Framework of the Proposed Genetic Algorithm

Based on the basic concept described in Subsection 3.1, each individual of GA consists of the value of the decision variable vector $\boldsymbol{x}_1$, and the selection, crossover, and mutation operations are applied to individuals. The selection requires fitness to evaluate an individual. In many existing GAs, it corresponds to the objective function value f. In contrast, in the proposed method, f is not calculated directly, and the fitness is calculated by using the estimation eg_{min}^k of the optimal objective function value g_{min}^k for each subproblem. The value eg_{min}^k is estimated by inputting the parameter $\boldsymbol{P}_{\boldsymbol{k}}$ into the neural network model trained in advance. From estimated values, the fitness ef is calculated by $ef = \sum_{k=1}^{q} eg_{min}^k$. Note that ef means the estimated value of $\min_{\boldsymbol{x}_2} f(\boldsymbol{a}_1, \boldsymbol{x}_2; \boldsymbol{P})$.

3.3 Dataset for the Neural Network Model

Our neural network model estimates the optimal objective function value g_{min}^k from the parameter $\boldsymbol{P}_{\boldsymbol{k}}$ of each subproblem. It is trained before the execution of the GA. Therefore, it is possible to create many training data in a long time to improve the estimation accuracy. The input of the model is the subproblem parameter, which is denoted by $\boldsymbol{SP}$. The output is the estimated value, which is denoted by s.

A sufficient number N of training data are created in the following way. First, an appropriate method generates N subproblem parameters $\boldsymbol{SP^i}$ $(i = 1, 2, \ldots, N)$. Next, the subproblem with each $\boldsymbol{SP^i}$ is solved, and let s^i be its optimal objective function value. The combination of $\boldsymbol{SP^i}$ and s^i is data, and the dataset is $\{(\boldsymbol{SP}^1, s^1), (\boldsymbol{SP}^2, s^2), \ldots, (\boldsymbol{SP}^N, s^N)\}$. For solving the subproblem, it is necessary to prepare an optimization method. Since the solution should be optimal, and the number of decision variables in each subproblem is smaller than that in problem (1), the exact methods such as the branch-and-bound method and the dynamic programming should be prepared. If preparing them is difficult, approximate methods such as the GA and the tabu search are used.

3.4 How to Determine the Final Solution

For problem (1) with decision variable vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the proposed GA determines the value of only $\boldsymbol{x}_1$ but never determines that of $\boldsymbol{x}_2$. Therefore, it does not yet determine the final values of all the decision variable vectors, that is, the final solution. To determine the final solution, after running the GA and determining the final value of $\boldsymbol{x}_1$ from the best individual, the optimization method prepared in Subsection 3.3 solves each subproblem. By finding the

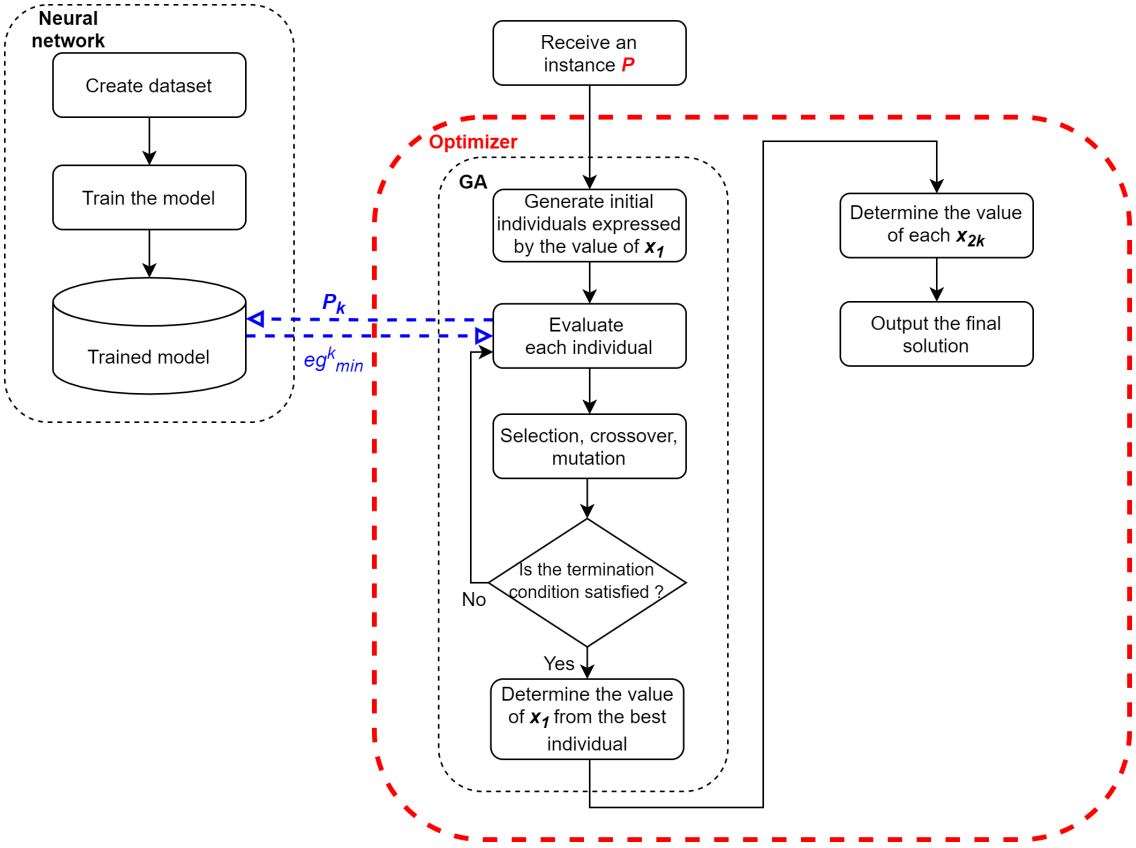

Figure 1: Flow of the whole proposed method.

optimal value of $\boldsymbol{x_{2k}}$ for each subproblem, $\boldsymbol{x}_2$ is determined, and the final solution is also determined.

Finally, the flow of the whole proposed method is summarized in Figure 1.

4 NUMERICAL EXPERIMENTS

This section shows experimental results of applying the method proposed in Section 3 and discusses its performance compared with other methods. As a case study, we use the drone problem described in Subsection 2.2.

4.1 Method

We use 20 instances in [11]. In these instances, the numbers of customers, drones, and takeoff points are, respectively, $n = 50, p = 6$, and $q = 10$. The flight time w_{ik} between each customer i and each point k is different depending on instances. To each instance, we apply the following four GAs:

- Proposed method: GA whose individual consists of the value of $\boldsymbol{x}_1$, and the individual is evaluated using the neural network model,
- GA1: GA whose individual consists of the values of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$,
- GA2: GA whose individual consists of the value of $\boldsymbol{x}_1$, and the individual is evaluated by determining the value of each $\boldsymbol{x_{2k}}$ using a greedy algorithm,
- GA3: GA whose individual consists of the value of $\boldsymbol{x}_1$, and the individual is evaluated by determining the value of $\boldsymbol{x}_2$ using a metaheuristic.

Each GA is executed 10 times. The experiments are conducted on a computer with Intel Core i7-7500U (2.70GHz).

4.2 Setting of the Proposed Method

This subsection describes the settings of the dataset, the neural network model, and the GA in the proposed method.

4.2.1 Setting of the dataset and the neural network model. First, we describe how to create the dataset $\{(\boldsymbol{SP}^1, s^1), (\boldsymbol{SP}^2, s^2), \ldots, (\boldsymbol{SP}^N, s^N)\}$. The number of training data is set to $N = 8000$. The input of the neural network model is the subproblem parameter vector $\boldsymbol{SP}$, and in the drone problem, it is a vector of flight times w_{ik} between one point k and customers to whom drones deliver from the point. Since $n = 50$, the dimension of $\boldsymbol{SP}$ is set to 50. However, mostly the number of customers in a subproblem is less than 50. For the subproblem, only some elements of $\boldsymbol{SP}$ are the flight times, and the remaining elements are set to 0. By randomly giving the number of customers and the flight times, each parameter vector $\boldsymbol{SP^i}$ of the dataset is generated. In

this generation, to improve the estimation accuracy, the elements of $\boldsymbol{SP^i}$ are sorted in descending order by $\boldsymbol{SP^i}$ and are normalized between 0 and 1 by the element. The subproblem with each $\boldsymbol{SP^i}$ is solved by the branch-and-bound method to obtain its optimal objective function value s^i. Not only training data but also 2000 test data are created in the same way.

Next, the architecture and parameters of the neural network model are as follows:

- Number of hidden layers: 1,
- Layer: Fully connected,
- Number of neurons: 50 (input layer), 50 (hidden layer), and 1 (output layer),
- Activation function: Rectified linear unit (ReLU),
- Loss function: Mean square error (MSE),
- Optimizer: Adam,
- Batch size: 512,
- Number of epochs: 1000.

4.2.2 Setting of the genetic algorithm. As described in Subsection 3.2, an individual is evaluated using the value eg^k_{min} estimated by inputting the parameter vector $\boldsymbol{P_k}$ of each subproblem into the neural network model. In this input, the elements of $\boldsymbol{P_k}$ are sorted in descending order, which is the same as those of parameter $\boldsymbol{SP^i}$ of the dataset. The fitness ef of the individual is calculated by $ef = \sum_{k=1}^{q} eg^k_{min}$. However, if the number of customers in the subproblem is less than or equal to the number of drones $p = 6$, it is optimal that each drone delivers to separate customers. In this case, the maximum flight time of drones equals the optimal objective function value g^k_{min} and is used instead of eg^k_{min}.

In the selection, tournament selection selects two individuals as parents. Uniform crossover or single point mutation is applied to them to generate two offspring. The generated offspring are replaced with the parents in the population. The parameter setting of GA is as follows:

- Population size: 10000,
- Number of generations: 80,
- Tournament size: 5,
- Crossover rate: 0.6,
- Mutation rate: 0.01.

4.3 Setting of the Other Methods

GA1 differs from the proposed method in the individual representation described in Subsection 4.1, the evaluation, and the mutation. In the evaluation of GA1, the objective function value f is directly calculated as a fitness. Single point mutation is applied to each part of an individual expressed by $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

GA2 and GA3 differ from the proposed method only in the evaluation. In the evaluation of GA2, the value of the decision variable vector $\boldsymbol{x}_{2\boldsymbol{k}}$ for each subproblem is determined by the greedy algorithm called the Longest Processing Time (LPT). LPT assigns the customer with the longest flight time to the drone with the shortest total flight time. Note that LPT is not general-purpose. An individual is evaluated by f calculated from the values of $\boldsymbol{x}_1$ and each $\boldsymbol{x}_{2\boldsymbol{k}}$. In the evaluation of GA3, the value of $\boldsymbol{x}_2$ is determined by the local search. In this local search, an initial candidate solution is generated randomly. Some new solutions are generated by changing each variable of the candidate solution randomly. If the best of the generated solutions is better than the candidate solution, it is replaced with the candidate solution. Those operations are repeated 10 times. An individual is evaluated by f calculated from the values of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

The parameter setting of GA1 and GA2 is the same as that of the proposed method, whereas that of GA3 differs in that the population size is 5500.

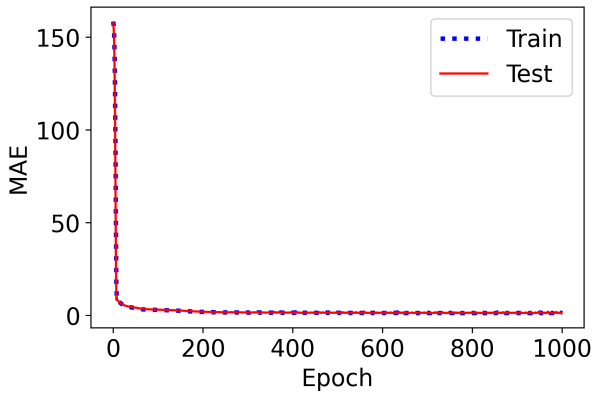

Figure 2: Training curves.

4.4 Results

First, we verify the estimation accuracy of the neural network model proposed in Subsection 3.2. Figure 2 shows the variation of the mean absolute errors (MAEs) for the training and test data. Both MAEs greatly decrease, and the MAE for the test data is 1.20 and slight at the final epoch. Therefore, the proposed model is properly trained.

Next, the proposed method is compared with GA1, GA2, and GA3. Table 1 shows the average objective function values $\bar{f}_{pm}$, $\bar{f}_{GA1}$, $\bar{f}_{GA2}$, and $\bar{f}_{GA3}$, respectively, by applying the proposed method, GA1, GA2, and GA3 10 times. Table 2 shows the average computation time $\bar{T}_{pm}$, $\bar{T}_{GA1}$, $\bar{T}_{GA2}$, and $\bar{T}_{GA3}$. These tables include results by an integer programming solver [11]: f_{IP} and T_{IP}. In [11], the computation is limited to about 3600 seconds. Figure 3 shows two average convergence curves of the proposed method executed for the instance 1 five times. One is the best of the estimated objective function values of all individuals, and the other is the true value for the estimation. Figure 4 shows four average convergence curves of the proposed method and other GAs for instance 3 five times.

We conduct the t-test between $\bar{f}_{pm}$ and each of $\bar{f}_{GA1}$, $\bar{f}_{GA2}$, and $\bar{f}_{GA3}$ in Table 1. The p-values for the three t-tests are 3.60×10^{-14}, 0.15, and 2.68×10^{-7}. Therefore, at a 5% significance level, the difference between $\bar{f}_{pm}$ and each of $\bar{f}_{GA1}$ and $\bar{f}_{GA3}$ is significant, but the difference between $\bar{f}_{pm}$ and $\bar{f}_{GA2}$ is not significant. These results of the t-tests and Table 1 show that the proposed method finds better solutions than GA1 and GA3 and as good solutions as GA2. Table 2 shows that the average computation time of the

Table 1: Comparison of objective function values by the proposed method and other GAs

Instance	$\bar{f}_{pm}$	$\bar{f}_{GA1}$	$\bar{f}_{GA2}$	$\bar{f}_{GA3}$	f_{IP} [11]
1	101.8	124.4	102.0	104.5	96
2	87.8	102.0	86.3	89.2	84
3	90.4	108.6	88.2	96.5	85
4	91.2	111.4	90.5	98.4	88
5	92.2	113.0	93.1	97.3	88
6	90.1	104.4	88.9	93.7	86
7	88.2	103.4	86.5	95.6	83
8	99.9	120.4	100.1	104.0	93
9	88.3	109.2	89.6	94.6	88
10	96.2	118.9	97.2	104.2	94
11	89.8	106.0	89.8	88.4	82
12	94.7	117.2	94.7	100.8	92
13	99.9	120.9	98.6	106.1	96
14	85.7	103.0	85.3	88.8	84
15	102.5	122.6	103.7	103.8	95
16	95.3	128.7	95.9	103.0	93
17	102.4	129.2	101.5	105.3	96
18	87.0	108.3	85.4	97.2	84
19	97.4	124.3	97.2	105.6	93
20	100.9	127.0	100.2	111.2	96
Average	94.085	115.145	93.735	99.41	89.8
Standard deviation	5.562	9.019	5.955	6.277	4.946

Table 2: Comparison of computation time [s] by the proposed method and other GAs

Instance	$\bar{T}_{pm}$	$\bar{T}_{GA1}$	$\bar{T}_{GA2}$	$\bar{T}_{GA3}$	T_{IP} [11]
1	16.21	14.61	14.72	275.72	3610.89
2	15.96	14.51	14.48	274.76	2476.56
3	16.07	14.51	14.67	275.14	3612.25
4	15.84	14.53	15.79	275.15	3614.61
5	15.74	14.53	14.97	275.04	1099.56
6	15.99	14.62	14.63	275.14	2138.09
7	16.56	14.51	14.42	274.86	351.75
8	27.27	14.51	14.38	273.34	3609.88
9	33.18	14.50	14.27	271.10	3604.55
10	17.20	14.60	14.33	275.37	3605.33
11	16.45	14.65	14.43	278.72	3606.98
12	16.27	14.59	14.54	282.60	3610.52
13	48.05	14.33	14.51	274.05	3610.59
14	16.15	14.44	14.25	273.56	3606.17
15	16.16	14.46	14.49	282.09	3607.63
16	16.27	14.32	14.30	280.79	3606.31
17	16.80	14.66	14.30	276.84	3612.31
18	16.29	14.46	14.51	278.17	261.53
19	16.53	14.39	14.37	287.40	3605.69
20	16.88	14.48	14.33	276.41	3610.33
Average	19.294	14.511	14.535	276.813	3028.077
Standard deviation	7.863	0.093	0.336	3.746	1111.137

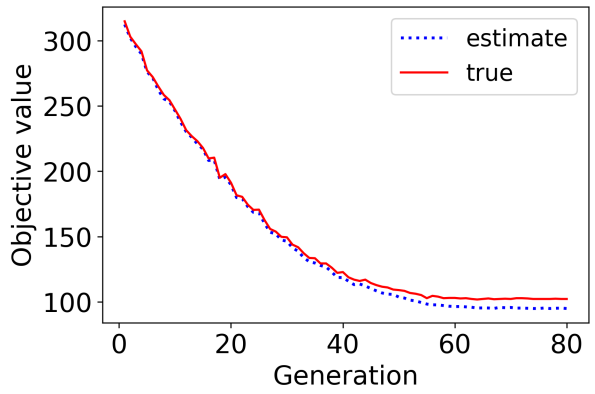

Figure 3: Convergence curves of the proposed method for instance 1.

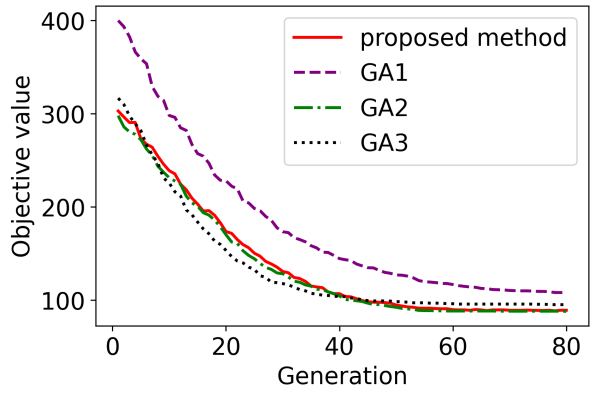

Figure 4: Convergence curves of the proposed method and other GAs for instance 3.

proposed method is about 257 seconds shorter than that of GA3. However, it is about 5 seconds longer than those of GA1 and GA2. Table 1 and 2 show also that the proposed method can find good solutions faster than the integer programming solver. Figure 3 shows that the proposed method gradually evolves the initial individuals with bad objective function values into individuals with good ones. Therefore, individuals are evolved correctly using estimations by the neural network model. Figure 4 shows that the proposed method and GA2 are slightly worse than GA3 until about the 45-th generation. GA1 is the worse at any generation.

4.5 Discussion

The proposed method can find better solutions than GA1, as shown in Table 1. However, it takes about 1.5 seconds longer than GA1 for most instances, as shown in Table 2. This is because the elements of parameter vector $\boldsymbol{P_k}$ in each subproblem are sorted in descending order in inputting $\boldsymbol{P_k}$ to the neural network model, as described in Subsection 4.2.2. Moreover, the proposed method takes much longer than GA1 for instances 8, 9, and 13. This is because the branch-and-bound method for determining the final solution takes longer if the subproblem it solves has many decision variables. For these same reasons, the proposed method takes longer than GA2.

The proposed method can find better solutions than GA1 and GA3 and as good ones as GA2, as shown in Table 1. Hybrid strategies such as GA2 and GA3 are often adopted for problems with multiple decision variable vectors. Compared to GA2, the proposed method does not use an algorithm depending on a problem, so it is general-purpose. Compared to GA3, the proposed method does not long search for the optimal solution of a subproblem, so it is not time-consuming. Another possible strategy is to use a GA whose individual consists of all decision variables like GA1. Compared to GA1, the proposed method can find much better solutions. Therefore, the proposed method is superior to methods employing other strategies.

Figure 3 shows that the estimated objective function value is almost the same as the true one until about the 40-th generation. However, from the generation, the estimated value is meaningfully smaller than the true one. This is because the estimation by the neural network model includes an error, and overestimated individuals survive.

5 CONCLUSION

This paper has proposed a GA with a neural network model to estimate optimal objective function values of subproblems. Experimental results for the drone problem show that the proposed method is superior. However, since its performance is evaluated only for the drone problem, we need to conduct experiments for other problems to verify its effectiveness.

Subproblems of problem (1) are similar to each other. However, there are cases where subproblems are not similar. For such a problem, multiple neural network models are required for the respective subproblems. We need to address the problem and develop a GA with the multiple neural network models.

ACKNOWLEDGMENTS

This work was partly supported by JSPS KAKENHI Grant Number JP20K11988.

REFERENCES

[1] D. E. Goldberg and J. H. Holland. 1988. Genetic Algorithms and Machine Learning. Machine Learning 3, 2 (October 1988), 95–99. https://doi.org/10.1023/A:1022602019183
[2] R. Said, M. Elarbi, S. Bechikh, and L. B. Said. 2021. Solving combinatorial bi-level optimization problems using multiple populations and migration schemes. Operational Research (January 2021). https://doi.org/10.1007/s12351-020-00616-z
[3] A. Peiravi, H. R. Mashhadi, and S. H. Javadi. 2013. An optimal energy-efficient clustering method in wire less sensor networks using multi-objective genetic algorithm. International Journal of Communication Systems 26, 1 (January 2013), 114–126. https://doi.org/10.1002/dac.1336
[4] W. Yu, C. Ha, and S. Park. 2021. A Hybrid Genetic Algorithm for Integrated Truck Scheduling and Product Routing on the Cross-Docking System with Multiple Receiving and Shipping Docks. Mathematical Problems in Engineering (March 2021). https://doi.org/10.1155/2021/2026834
[5] H. You, Z. Pan, N. Liu, and X. You. 2020. User Clustering Scheme for Downlink Hybrid NOMA Systems Based on Genetic Algorithm. IEEE Access 8 (July 2020), 129461–129468. https://doi.org/10.1109/ACCESS.2020.3009018
[6] Y. -Y. Tan, Y. -X. Li, and R. -X. Wang. 2020. Scheduling Extra Train Paths Into Cyclic Timetable Based on the Genetic Algorithm. IEEE Access 8 (May 2020), 102199–102211. https://doi.org/10.1109/ACCESS.2020.2997777
[7] E. Vin, P. D. Lit, and A. Delchambre. 2005. A multiple-objective grouping genetic algorithm for the cell formation problem with alternative routings. Journal of Intelligent Manufacturing 16 (April 2005), 189–205. https://doi.org/10.1007/s10845-004-5888-4

[8] Y. Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation 1, 2 (June 2011), 61–70. https://doi.org/10.1016/j.swevo.2011.05.001

[9] H. L. Le, D. Landa-Silva, M. Galar, S. Garcia, and I. Triguero. 2020. A Hybrid Surrogate Model for Evolutionary Undersampling in Imbalanced Classification. Proceedings of IEEE Congress on Evolutionary Computation, Glasgow, UK. https://doi.org/10.1109/CEC48606.2020.9185774

[10] Z. Tan and H. Wang. 2020. A Kriging-Assisted Evolutionary Algorithm Using Feature Selection for Expensive Sparse Multi-Objective Optimization. Proceedings of IEEE Congress on Evolutionary Computation, Glasgow, UK. https://doi.org/10.1109/CEC48606.2020.9185825

[11] K. Mishima and Y. Karuno. 2019. A Parcel Delivery Scheduling Problem with Multiple Unmanned Aerial Vehicles and a Single Truck. Proceedings of the Scheduling Symposium, 19–24.