Genetic Algorithm with Gene Regulatory Networks based Optimization Method for Distributed Video Analysis System

Seishiro Inoue
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
s-inoue@ist.osaka-u.ac.jp

Masaaki Yamauchi
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
m-yamauchi@ist.osaka-u.ac.jp

Daichi Kominami
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
d-kominami@ist.osaka-u.ac.jp

Hideyuki Shimonishi
Cyber Media Center, Osaka University
Osaka, Japan
shimonishi.cmc@osaka-u.ac.jp

Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
murata@ist.osaka-u.ac.jp

***Abstract*—To construct a digital twin, we have to analyze a large amount of video data obtained from the real world in real-time to perceive an accurate current situation. The increase in the video traffic flowing through networks and the accompanying increase in power consumption are problems. One solution is edge computing to distribute video analyzing tasks; it requires dynamic control of task distribution and selecting analyzing models according to the network status and application requirements. Our research group has formulated an optimization problem for dynamic control of distributed video analysis systems and proposed a method to optimize power consumption using a genetic algorithm. However, as the scale of the problem expands, there are cases where the genetic algorithm cannot solve the problem because it falls into the local solution. Therefore, we focus on gene regulatory networks that map a genotype of an organism to its phenotype. The characteristics of gene regulatory networks are to memorize specific phenotypes, to mutate multiple phenotype bits from one-bit perturbation in genotype simultaneously, and to generate new phenotypes similar to the memorized ones by the network structure. In this study, we propose a dynamic control method for distributed video analysis systems by applying gene regulatory networks to genetic algorithms. We used a mathematical model of gene regulatory networks and made it memorized several optimal solutions as phenotypes. We confirmed the escape from local solutions by recalling the stored solutions and generating multiple phenotype bits mutations. We also verified the adaptability improvement to the dynamic network environment and a new application requirement by using the gene regulatory network.**

***Index Terms*—Adaptive evolution, digital twin, gene regulatory network, genetic algorithm, optimization problem.**

I. Introduction

Digital twin technology, which highly integrates a real and a virtual world, is attracting attention. The digital twin reproduces the same environment as the real world in the digital world as if it were a twin by collecting data about objects and environments using cameras or sensors. The digital twin enables continuous monitoring and optimization of objects and systems in the real world. The digital twin technology is expected to bring new value to a variety of industries [1].

To construct the digital twin, accurate and real-time recognition of the real-world situation is necessary. Therefore, we have to collect and analyze a large amount of video and sensor data in a short time. A large amount of video data is generated from multiple cameras or LiDAR sensors and collected on a cloud server via the network. Then, AI video analysis technologies, such as YOLO [2], process the data to construct the digital twin. Users can view the analyzed results from the digital twin [3]. Research of beyond 5G and 6G is now actively conducted; it is expected that high-speed, high-capacity communications with low latency will be realized. It causes development of applications including high-capacity communication in a short time; we can expect the rapid increase of traffic flow through network in the future. The traffic increase is expected to significantly increase power consumption in the network [4]. Then reducing power consumption of the entire network system is a major challenge in the utilization of the digital twin in beyond 5G and 6G era.

Edge computing technology is one of the solutions for the communication delay and power consumption by the increase in video traffic. Edge computing is that places servers, named edge servers, closer to the data source than the cloud servers and allocates part of the data processing of the cloud servers to the edge servers [5]. The power consumption of the network and communication delays are reduced because the amount of traffic in the entire network is reduced. The processing latency of the video data is also reduced because the edge servers are close to the terminals of users [6].

When we operate edge servers, we have to dynamically determine how processing is distributed according to the network status and types of applications. Network status includes network congestion and usage amount of edge and cloud

This work was supported by MIC under a grant entitled ”R&D of ICT Priority Technology (JPMI00316).”

servers. The required processing latency and accuracy of analysis vary depending on the applications. To achieve efficient distributed processing for lower power consumption, we have to monitor network status regularly and dynamically optimize the distribution of processing depending on the applications.

Our research group has formulated a power consumption optimization of a distributed video analysis system as a combinatorial optimization problem and proposed an optimization method using Genetic Algorithm (GA) [7]. The system consists of three types of computational resources, terminal, edge server, and cloud server, which are connected by networks. When a camera attached to the terminal acquires video data, an application on the terminal sends a request to analyze the video data to the optimization module with a maximum latency requirement and a minimum accuracy requirement. The optimization module determines the processing distribution ratio and video analysis AI model for each computing resource so that the power consumption is minimized while satisfying the two requirements. Note that, since the video analysis AI models have a trade-off between processing speed and analysis accuracy, we have to set appropriate AI models according to the requirements. Because the system has multiple computational resources and networks and the complexity increases exponentially concerning the size of the system, our group has solved it with GA. In addition, it has been proposed to use the Bayesian Attractor Model (BAM). The BAM stores multiple attractors of network status information with corresponding solutions to the problem. When the BAM judges a current network status as similar to the stored one, GA uses the corresponding solution to improve adaptation speed [8].

However, as the scale of the problem expands, GA cannot achieve optimal solutions and falls into local ones in some cases. This is because, while GA mutates one bit in many cases, the mutation is too small to escape from the local optima; and the mutated individuals are eliminated in the evolution process. In addition, as the problem scale and the information on the network status increases, it takes time to get all the information and judge the similarity of the network status. It may also be possible to misjudge if the network state is partially similar. Therefore, it is necessary to be able to memorize and recall past optimal solutions, independent from the information on the network state.

We focus on Gene Regulatory Networks (GRN), a mechanism in living organisms that maps the genotype of an organism onto a phenotype of a cell. One of the characteristics of GRN is that mapping from a wide genotypic space to a narrow phenotypic space facilitates the expression of specific phenotypes. It is also known that mutation on a particular gene in the genotype called a sensitive node, causes multiple changes of phenotypes simultaneously [9]. The modular structure of GRN also allows only a particular module to change, leading to generate new phenotypic combinations of past phenotypes.

In this paper, we propose a method to utilize GRN in GA for optimizing distributed video analysis systems against fluctuations in network conditions and application requirements. We confirm that GRN, an adaptive evolutionary mechanism of living organisms, improves the adaptive capability towards environmental changes surrounding the system. Specifically, by using the mathematical model of GRN that pre-learned optimal solutions for several requirements as phenotypes, we verify whether genotypic and phenotypic mutations improve the adaptability. We verify whether the GRN-based method can find the optimal solution in less time than GA by recalling the memorized phenotype without grasping the entire system state when the system state changes. Also, by simultaneously changing multiple phenotypes with a small number of genotypic mutations, it will be verified whether it is possible to escape from local solutions and quickly find an optimal solution. In addition, we evaluate whether the GRN-based method can adapt to inexperienced network status and application requests by utilizing memorized phenotypes or new phenotypes generated from the network structure of GRN.

II. Related work

A. Gene Regulatory Networks (GRN)

GRN is a network of gene interactions. The interactions can either promote or suppress activity. When genes regulate other's activity, various cells are expressed and vital organs are generated. By setting different initial genes, i.e., genotypes, it is possible to generate multiple types of cells and organs from the same GRN. The expression pattern of a gene that is generated by GRN is called phenotype.

In biological evolution, only mutations that are advantageous for survival are retained by natural selection. Thus, GRN may be biased in the distribution of phenotypes through the evolutionary process. Such biases are reported to have three characteristics in [10]. (1) Distributed associative memory capable of storing and recalling past phenotypes. (2) Associative memory that accurately represents a complete phenotype from a partial embryo. (3) Generalization that generates new combinations of phenotypic features. In addition, it is possible to mutate multiple phenotypes simultaneously by mutating specific genes of the genotype, called a sensitive node, to give a large variation in the phenotype; this character is verified using boolean networks, one of the models of GRN [9]. In a combinatorial optimization problem for a system subject to environmental variation such as the one targeted in this paper, adaptability may be enhanced by recalling past memories from partial environmental conditions. Furthermore, if phenotypic generalization is utilized, it is expected to generate appropriate phenotypes for unknown environmental conditions.

We use a mathematical $n \times n$ adjacency matrix model [11] as GRN. The n bit phenotype is calculated by multiplying the n bit genotype for multiple times. Since the distribution of phenotypes is biased according to the matrix, the specific phenotypes can be expressed multiple times as memory.

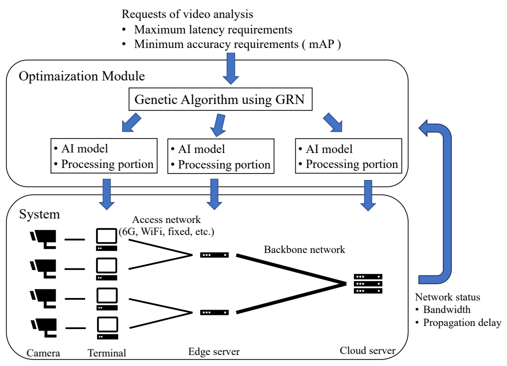


Fig. 1. System model.

B. Energy optimization of distributed video processing system with BAM [8]

Our research group formulated the energy optimization of a distributed video analysis system as a combinatorial optimization problem and proposed a method to solve it by a GA with BAM. The method with BAM assesses the similarity of the network environment using all the network state information. However, it is likely to make misjudgments when the network state is partially similar as the system scale becomes large. In this study, we propose an optimization method that does not use network state information for similarity judgments. We use GRN, which can generate similar phenotypes as solutions without network state information.

III. Overview of the System Models and the Proposed Method

A. System overview

We describe the system overview in Fig. 1. The system has three computation resources, terminals connected to cameras, edge servers, and cloud servers. The terminals and the edge servers are connected by access networks. The edge servers and the cloud servers are connected by backbone networks. An application on a terminal sends a request to the system with requirements of latency and accuracy. An optimization module monitors the information of each network, such as bandwidth and propagation delay. When the module receives the video processing request, it creates a session on the path via the terminal, an edge server, and a cloud server. Then, the module allocates the appropriate processing ratio and AI model for each computing resource using our proposed method, "Genetic Algorithm using GRN" in Fig. 1. After the allocation, video data is sent from the camera to the terminal and the edge server according to the processing ratio; the cloud server processes the remaining data. Note that, the optimization module does not supply the video analysis task until it finds a solution that satisfies the requirements of the request.

B. Definition of optimization problem

The energy optimization of the distributed video analysis system is defined as a combinatorial optimization problem. We use the almost same definition of previous work [8] of our research group. The proposed method optimizes the processing ratio and AI model that satisfies the application requirement and reduces the power consumption of the entire system. The processing ratio is the ratio of allocated video analysis tasks for each computing resource. The AI model is the model of the video analyzer to be used in each computing resource. Two requirements are processing delay requirements and accuracy requirements of video analysis. Note that, since the video processing AI models have a trade-off between processing speed and analysis accuracy, we have to set appropriate models according to the requirements.

1) Optimized variables: The two types of control information are processing ratio W_s^d and AI model M_s^d, where s denotes session and d denotes device. The optimal combination of these variables considering all sessions and the entire power consumption in the system is searched.

2) Constraints: The constraints in this optimization problem are the end-to-end (E2E) delay and analysis accuracy of the processing request.

The E2E delay is defined as

$$T_s(t) = \sum\nolimits_{d \in D^s} T_s^d(t) + \sum\nolimits_{n \in N^s} T_s^n(t) \le T_s^{max}, \forall s \in S, \tag{1}$$

where S is the set of all sessions. D^s and N^s are the set of devices and networks used by session s, respectively. $T_s^d(t)$ is the processing delay for video analysis on device d of session s, and $T_s^n(t)$ is the processing delay and the transmission delay in network n of session s. The constraint of the processing delay is $T_s(t) < T_s^{max} \forall s \in S$, where T_s^{max} is the upper limit of processing delay sent from the application.

The analysis accuracy is defined as.

$$A_s = \sum\nolimits_{d \in D^s} A^{M_s^d} W_s^d / |D^s| \ge A_s^{min}, \forall s \in S, \tag{2}$$

where A_s is the video analysis accuracy in session s, $A^{M_s^d}$ is the analysis accuracy when using model M_s^d on device d, and $\sum_{d \in D^s} A^{M_s^d} W_s^d / |D^s|$ is a weighted average of analysis accuracy. The constraint of the analysis accuracy is $A_s(t) > A_s^{min} \forall s \in S$, the A_s^{min} is the lower limit of analysis accuracy sent by the application.

3) Objective function: The overall system power consumption $E(t)$ at time t is an objective function. It is defined as the sum of the power consumption of devices $E^d(t)$ and of network $E^n(t)$ of each device d and network n as

$$E(t) = \sum\nolimits_{d \in D^s} E^d(t) + \sum\nolimits_{n \in N^s} E^n(t). \tag{3}$$

C. Optimization method

1) Encoding: To solve the optimization problem, the control information for distributed processing needs to be encoded. The phenotype is represented in a bit string as $[W_s^d \forall s \in S, \forall d \in D^s, M_s^d \forall s \in S, \forall d \in D^s]$, where session s and computing resource d.

2) Genetic Algorithm (GA): GA is an optimization method that performs genetic manipulations such as crossover and mutation on a population containing N individuals and conserves the individuals suitable to the environment for the next generation. The suitability is calculated by the fitness function, defined in (4). Each individual has a genotype and phenotype; and in GA, the phenotype is mapped directory from the genotype. Individuals are selected according to their fitness, and individuals similar to them are generated through genetic manipulation to search the solution space around the similar one. The procedure of a simple model of GA, called Simple GA (SGA) is below.

i To initialize genotypes of N individuals to random values.
ii To generate new individuals by crossover and mutation.
iii According to the fitness of the phenotype mapped by the genotype, selecting N individuals remaining in the next generation, and repeat i) and ii).

3) Fitness function: The fitness function indicating adaptability of individual is defined as

$$F(t) = -E(t) - \alpha \sum_{s \in S} (T_s(t)^{max} - T_s) - \beta \sum_{s \in S} (A_s(t) - A_s^{min}). \quad (4)$$

The fitness is high with the reduction of the power consumption $E(t)$. The second and third terms are penalties of delay and accuracy respectively, where α and β are coefficients.

D. Proposed method to use GRN for GA

We embed GRN in GA. We deploy an adjacency matrix as GRN in each individual and map the phenotype through the GRN.

1) Calculation of phenotype: We calculate the phenotype array by multiplying the genotype array and the adjacency matrix of GRN multiple times. The phenotype represented by $P(t) = [p_0(t), p_1(t), \ldots, p_n(t)]$ at k-th step is calculated as

$$p_i(k+1) = p_i(k) + \tau_1 \sigma \left(\sum_{j=0}^{n} b_{ij} p_j(k) \right) - \tau_2 p_i(k), \quad (5)$$

where genotype vector is $G = [g_0, g_1, \ldots, g_n](-1 \leq g_k \leq 1)$, GRN matrix is $B = [b_{ij}](0 \leq i, j \leq n)$, and $p_i(0) = g_i$. The fitness of an individual is calculated by $p_i(k^*)$, where k^* is a parameter of calculation steps. τ_1 and τ_2 are coefficients of an effect of interaction between genes and a decay ratio of the expression pattern respectively. σ is a sigmoid function and we set a hyperbolic tangent function.

2) Algorithms: We introduce phenotype mutation in addition to the genotype mutation. The phenotype produced by GRN is biased. Therefore, individuals biased by GRN with only genotype mutation cannot generate every phenotype in the phenotypic space. By adding the phenotype mutation, the proposed method can achieve every solution. The balance of each mutation is configured by the parameter N_2. The algorithm of the proposed method works in the following steps and repeats ii), iii), and iv).

i To initialize genotypes of N individuals to random values.
ii To calculate the fitness from the phenotype of individuals and save the best N_1 ones as elite.
iii By phenotypic mutation, generating N_2 individuals for the next generation from the current population.
  a) Selecting one individual by selection method, such as roulette selection.
  b) To generate a new individual by applying mutation and crossover to the phenotype of the selected one.
  c) To repeat a) and b) until N_2 individuals are generated.
iv By genotypic mutation, generating $N - N_1 - N_2$ individuals for the next generation from the current population.
  a) To select one individual by the selection method.
  b) Generating a new individual by applying mutation and crossover to the genotype of the selected one and calculating its phenotype.
  c) To repeat a) and b) until $N - N_1 - N_2$ individuals are generated.

3) Memorizing phenotypes by GRN: We refer [10] to design an adjacency matrix of GRN to express specific phenotypes. We evolve GRN by selection pressure in the direction closer to the target phenotypes to be memorized. First, we initialize the genotype vector G and the GRN adjacency matrix B with zero. Second, we select one target phenotype and mutate G and B to minimize the hamming distance between the generated phenotype by GRN and the target one. Third, we switch the target phenotype and continue mutations. Switching the target phenotype multiple times creates a bias in the expression of the phenotype, i.e., GRN memorizes the phenotypes.

IV. Evaluation

We confirm that the GRN-based proposed method improves adaptability to the dynamic environment and how the characteristics of GRN affect it. The adaptability is measured by the fitness function $F(t)$ defined as (4) and power consumption $E(t)$ in time t defined as (3).

The environmental changes in this distributed video analysis system include two cases: switching applications that use the system and fluctuating network status. The applications are divided into two types: applications that have used the system multiple times and applications that have never used the system or used the system a few times. GRN can memorize optimal solutions that can satisfy the requirements of the former, calling it a known application, and achieving a reduction of power consumption. When the known application sends a request, the memory stored in the GRN can make the adaptability high. Optimal solution of the latter, calling it a new application, is not memorized in GRN. In addition, it is considered the network status, such as the available bandwidth for this system, changes more frequently than the application switching. Thus, we simulate three cases to evaluate the adaptability of the GRN-based proposed method: the application switches to the known ones, the known application and dynamic network bandwidth, and switching to the new application in dynamical network bandwidth.

In this optimization problem, the structure of the optimal solution can be clustered into several patterns. Concretely, the distribution ratio of tasks is distinctive, such as processing all video analysis tasks in the cloud server or processing some tasks at the edge and most tasks at the terminal. The switching of the application requirements tends to reform the structure of the solution, i.e., requiring large-scale phenotypic mutations, such as changes in the processing proportions and deployed AI models. On the other hand, differences in network bandwidth rarely vary the structure of the solution greatly. The changes affect mainly the AI model selection and some processing proportions. In other words, it causes smaller changes than the application switching. Based on these characteristics, we discuss the utilization of GRN with environmental changes.

A. Simulation settings

We use three types of devices ($D^s = \{t, e, c\}$): terminal, edge server, and cloud server, and two types of communication networks ($N^s = \{n_{access}, n_{backborn}\}$): access network and backbone network. We assume that one terminal executes one application and constitutes one session. The simulation environment is configured for the cases with one, two, and four sessions. In this paper, we focus on cases of four sessions due to page restriction. The number of computing resources and structure of the network is shown in Fig. 1.

1) Device setting: Table I shows the configures of each device. We use an estimation model based on actual measurements [8] to define the value of the processing latency and power consumption model.

GPU processing load rate $L^d(t)$ is estimated as

$$L^d(t) = \left(\sum\nolimits_{s \in S^d} O_s^d(t)\right) / \left(C^d \mathit{Eff}^d\right), \qquad (6)$$

where d and t are the device and the time respectively. Processing load $O_s^d(t)$ on session s is calculated as

$$O_s^d(t) = \left((O^{M_s^d(t)} + O^A) W_s^d(t) + O^B\right) (\text{FPS}). \qquad (7)$$

By using (7), processing latency $T_s^d(t)$ and power consumption $E^d(t)$ is calculated as

$$T_s^d(t) = \left(O^{M_s^d(t)} + O^A + O^B\right) / \left(C^d \mathit{Eff}^d L^d(t)\right), \qquad (8)$$

$$E^d(t) = E^d_{\mathit{IDLE}} + \alpha^E E^d_{\mathit{CPU\text{-}TDP}} + \beta^E E^d_{\mathit{GPU\text{-}TDP}} L^d(t). \qquad (9)$$

2) Network settings: We set the network of simulations as shown in table II unless there is an environmental change. Calculations formula and models are based on the previous work [8]. The network transmission delay $T_s^n(t)$ and network power consumption $E^n(t)$ are defined as

$$T_s^n(t) = R\left(1 - \sum\nolimits_{d \in D^{s,n}_{pass}} W_s^d(t)\right) / B^n(t) + T^n_{prop}, \qquad (10)$$

$$E^n(t) = \sum\nolimits_{s \in S} R\left(1 - \sum\nolimits_{d \in D^{s,n}_{pass}} W_s^d(t)\right) E_{bit}, \qquad (11)$$

where $D^{s,n}_{pass}$ is the set of devices that a session s has passed before reaching network n and $B^n(t)$ is the available bandwidth on network n at time t satisfying $B^n(t) \leq B^n_{max}$.

TABLE I
DEVICE SPECIFICATIONS AND ESTIMATED PARAMETERS.

Spec and *variables*	Terminal t	Edge server e	Cloud server c
CPU	Core i7-8700T	Xeon GOLD 6226R ×2	Core i9-10940X
TDP $E^d_{\mathit{CPU\text{-}TDP}}$	35W	150W ×2	165W
GPU	GeForce GTX1070	Tesla T4 ×2	RTX A5000
FP32 C^d	6.463Tflops	8.141Tflops×2	27.77Tflops
TDP $E^d_{\mathit{GPU\text{-}TDP}}$	150W	70W ×2	230W
Efficiency Eff^d	0.4	0.39	0.48
IDLE power E^d_{IDLE}	55.236W	334W	258.3176W
FLOPS A and B (O^A, O^B)	(1.5, 0.1)	(2.2, 0.1)	(8.1, 0.4)
(α^E, β^E)	(0.97, 0.78)	(0.97, 0.67)	(0.29, 0.81)

TABLE II
NETWORK PARAMETERS AND SET VALUES IN THE SIMULATIONS.

Network type	n_{access}	$n_{backborn}$
Bandwidth: B^n_{max}	25 Mbps	250 Mbps
Propagation delay: T^n_{prop}	10 ms	10 ms
Power consumption per bit: E_{bit}	193×10^{-9} W	60×10^{-9} W
Bit per frame: R	470 Kbit	

3) Predefined application performance requirements: We set three types of requirements as the known applications. Their optimal solutions are memorized as phenotypes by GRN. Table III shows the predefined performance requirements and the phenotypes. The phenotypes are obtained by multiple searches using SGA because the computation of all bit-string combinations requires an enormous time. In this search, the network bandwidth is fixed to the value in Table II, which is an almost mean value of the network bandwidth variation.

4) Settings of parameter, etc.: In the proposed method, unless we specified, $N = 100$, $N_1 = 5$, and $N_2 = 50$. In other words, the population has 100 individuals and the number of individuals generated by the proposed method is $5 : 45 : 50$ for elite individuals, generated by genotypic mutation, and generated by phenotypic mutation.

In genetic algorithm, we introduce roulette selection and

TABLE III
PREDEFINED PERFORMANCE REQUIREMENTS AND ITS SOLUTIONS.

App. requirement for each session	Latency T_s^{max}	Accuracy A_s^{min} (mAP)	Memorized phenotype	Fitness
A1	0.04238 s	26.855 %	[0,0,6,4,6,2,	-365.696
A2	0.12856 s	20.926 %	0,0,3,2,2,4,	
A3	0.11831 s	27.375 %	0,0,6,7,6,6,	
A4	0.10322 s	22.440 %	0,0,5,1,3,7]	
B1	0.06481 s	34.535 %	[0,0,5,6,7,5,	-389.289
B2	0.05974 s	41.178 %	0,0,6,7,0,5,	
B3	0.04144 s	26.532 %	6,0,0,0,7,3,	
B4	0.09141 s	37.337 %	0,0,5,1,2,4]	
C1	0.12314 s	48.583 %	[0,0,7,0,0,4,	-379.944
C2	0.03937 s	31.529 %	7,0,0,0,0,2,	
C3	0.12405 s	33.166 %	0,0,5,3,7,4,	
C4	0.11723 s	23.430 %	0,0,5,1,3,0]	

TABLE IV
SPECIFICATIONS OF VIDEO PROCESSING AI MODELS [12].

Model	Accuracy: $A^{M_s^d}$	FLOPS: $O^{M_s^d(t)}$	M_s^d
YOLOv3-tiny	33.1 %	5.6 B	$\{0,1,2\}$
YOLOv3	55.3 %	65.9 B	$\{3,4,5\}$
YOLOv3-spp	60.6 %	141.5 B	$\{6,7\}$

elite survival strategy, two-point crossover with crossover rate of 0.2, and point mutation. SGA also stores 5 % of elites.

We set the range of W_s^d and M_s^d as 0–7, the latter's corresponding models are described in Table IV. We also set $(\alpha,\beta)=(10^5,10^5)$, $(\tau_1,\tau_2)=(1.0,0.2)$ and $k^*=10$.

B. Simulation of application switching to the known ones

1) Evaluation procedures: We set the application requirements from the predefined set shown in the table III as follows: requirement A→B→C→A→.... and set it to change periodically every 60 s. We perform 10 trials with 10 application changes, in other words, 90 switches excluding the first one of each trial. We compare the actual computation time to adapt to the environmental change and power consumption for both the proposed method and SGA.

2) Results: We show the transition of fitness in Fig. 2.

The proposed method can find the optimal solution in a shorter time than SGA after environmental changes. The average computation time and generation of GRN is 0.56 s and 25, and of SGA is 45 s and 2,897 respectively. This is because GRN recalls the memorized phenotypes in a few generations. GRN achieves shorter computation times, although it requires a longer computation time of one generation than SGA due to the phenotype computation. Also, out of 90 environmental changes, SGA found the optimal solution 28 times, whereas GRN found it in all cases. In the proposed method, a single genotype mutation allows the simultaneous mutation of multiple phenotype bits and avoids falling into a local solution.

When the number of sessions is 1, we confirm that the SGA also can reach the optimal solutions in a short time for all environmental changes. This is because the bit length of the genotype is short and the optimal solution can be reached with a relatively small number of mutations in the case. We have checked that the increase in sessions makes the problem difficult and SGA could not solve it.

To evaluate the power consumption, we first compare the unavailable time that the system cannot provide the video analysis service because the performance requirements are not satisfied. As an average time for 10 trials, the total unavailable time was 171 s in SGA, whereas it was 1.27 s in the proposed method. We can confirm that the stability of the service was greatly improved. A comparison of the average power consumption while the performance requirements are satisfied shows that SGA is 505 W while the proposed method is 378 W. It indicates a significant reduction in power consumption.

We also focus on the parameter N, the number of 2 individuals generated by the phenotypic mutation. When the N2 is small, such as 15, the search time is shorter than N2 = 50 case because many individuals are generated by genotypic mutations, i.e., recall. On the other hand, in N2 = 85, it takes longer time to find the optimal solution than in N2 = 50 case because few individuals are generated by recalling. We confirmed that phenotype switching requires changing the solution structure and genotypic mutation, which changes multiple phenotype bits simultaneously, is effective.

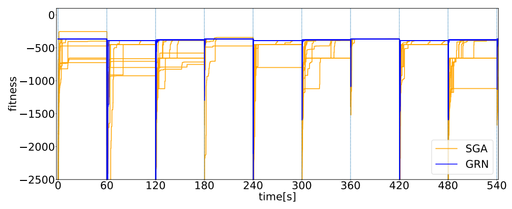


Fig. 2. Fitness transition when the requirement changes every 60 seconds. We show results of 10 trials of the proposed method (GRN) and SGA.

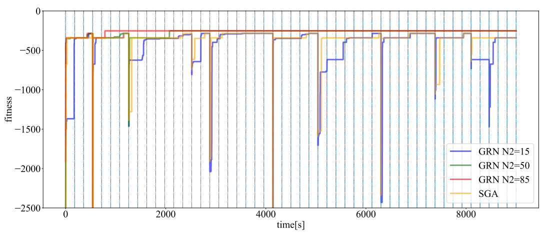
C. Simulation of dynamic network bandwidth with the known application

We confirm the adaptability of GRN in dynamic network bandwidth fixing the applications to known ones. Although a change in network bandwidth does not require a significant change in the structure of the optimal solution, it does require changes in some parts of the solution. Namely, in this section, we evaluate the adaptability in similar environments to the ones that GRN memorized.

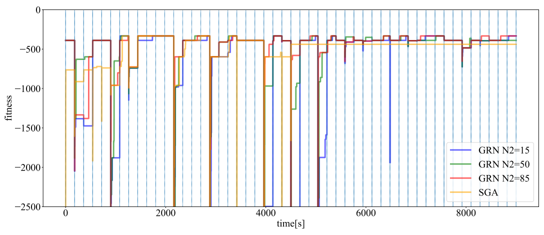
1) Evaluation settings: We randomly select the bandwidth of the access network and the backbone network in the range of 10–50 Mbps and 10–500 Mbps respectively to make 50 patterns. We deploy in order of the patterns each 180 s. We perform it in 10 trials. Thus, the 500 environment is simulated in each requirement. The application requirement is fixed in each trial selected from the known ones shown in Table III.

2) Results: Fig. 3 shows the time transition of the fitness when the application requirement is set to requirements A, B, and C in Table III, respectively.

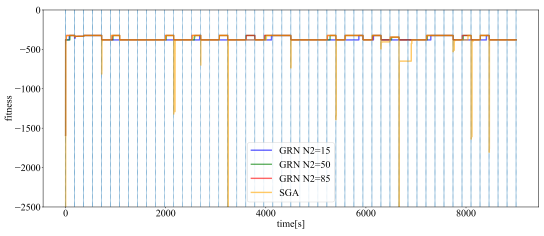
In each requirement, SGA and GRN show different transitions. This is because the magnitude of changes in solution space by the bandwidth fluctuations depends on the constraints. When the constraint is less strict, the solution space change tends to be small and the transition tends to be stable. In requirement A with less strictness of accuracy constraints, there are less susceptible to network fluctuations because the tasks tend to be processed by the terminals. In requirement C with less strictness of delay constraints, because the bandwidth changes are tolerated, the solution space is not changed largely. On the other hand, when the application requires strict constraints, such as requirement B, the solution space sensitively changes. This is because tasks need to be processed in high-performance servers, like the cloud, due to the accuracy constraints, thus the effect of bandwidth becomes large. Moreover, the solution tends to violate the delay constraints when the bandwidth changes.



(a) Performance requirement is fixed at requirement A.

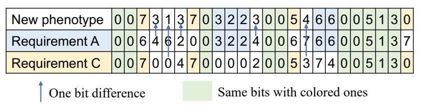


(b) Performance requirement is fixed at requirement B.



(c) Performance requirement is fixed at requirement C.

Fig. 3. The fitness transitions of the proposed method (GRN) and SGA with changing bandwidth in 50 patterns. The vertical dotted line is changed time.



New phenotype	0	0	7	3	1	3	7	0	3	2	2	3	0	0	5	4	6	6	0	0	5	1	3	0
Requirement A	0	0	6	4	6	2	0	0	3	2	2	4	0	0	6	7	6	6	0	0	5	1	3	7
Requirement C	0	0	7	0	0	4	7	0	0	0	0	2	0	0	5	3	7	4	0	0	5	1	3	0

One bit difference
Same bits with colored ones

Fig. 4. New phenotype generated by the combination of memories.

In the case of requirement A, 10 s after the environmental change, GRN, where $N_2 = 50$, satisfies the requirement 489 times out of 500, and SGA satisfies it 453 times. GRN achieves higher fitness 334 times, SGA achieves higher fitness 39 times, and the same fitness 163 times. The GRN achieves higher adaptability than SGA. In particular, GRN, where $N_2 = 50$, reaches an adaptable solution to most network bandwidths in 8 out of 10 trials, such as the solution of 12 th environment in Fig. 3(a). However, SGA cannot reach it. This solution might be a local one, but no better solution was found in each trial. This is because GRN-based individuals can search around the solution of requirement A more widely than SGA by multi-bit phenotype mutations from single-bit genotype mutation. Another reason is the characteristic of GRN that generates a new phenotype from memorized phenotypes by the network structure. A phenotype shown in Fig. 4 is generated by GRN and it is similar to the solution of requirements A and C. The phenotype shows high fitness in requirement A. GRN can search around such a new phenotype.

In Fig. 3(a), the unavailable time is 149 s for GRN and 382 s for SGA; the average power consumption is 288 W in GRN and 366 W in SGA. Power consumption is reduced and the service is provided stably.

In the case of requirement B, there are 371 cases where the fitness of GRN, where $N_2 = 50$, after 10 s from network changes is higher than SGA, 89 cases where SGA is higher than GRN, and 40 cases where they achieve the same fitness out of 500 cases. Thus, the proposed method has higher adaptability than SGA in the same environment. SGA tends to fall into local solutions in the case of requirement B. For example, the SGA in Fig. 3(b) finds it in the 26 th environment. This happened in 9 out of 10 trials. On the other hand, GRN does not fall into the local solution and it leads to finding a higher fitness solution than SGA. GRN can escape from the local solution by multi-bit phenotype mutation caused by single-bit genotype mutation.

In some cases, GRN takes longer to satisfy the requirements. The number of times requirements are satisfied after 10 s from the environmental change is 408 out of 500 cases for GRN and 467 for SGA, and after 180 s, it is 446 for GRN and 492 for SGA. One reason is that the local solution found by SGA was the common solution to satisfy the requirement. Another reason is that the solution memorized by GRN was not adaptable to the bandwidth changes. We have to select an adaptable solution to be memorized in different network bandwidths, which is our future work. In a comparison of power consumption, the average power consumption of GRN and SGA is 389 W and 455 W respectively in Fig. 3(b). However, the unavailable time of GRN is $1,294$ s which is longer than SGA, 296 s. This is a case in which GRN could not achieve more adaptability than SGA.

In the case of requirement C, GRN achieves higher adaptability than SGA. In 500 simulations, GRN is superior in the number of times that requirements are satisfied after 10 s from environmental changes: 500 times for the GRN and 461 times for the SGA. Also in Fig. 3(c), the unavailable time is 246 s for GRN and 257 s for SGA, which indicates GRN can provide stable service. The average power consumption is 392 W for GRN and 421 W for SGA, we can find that power consumption is reduced. The memorized solution has a high fitness even if the bandwidth changes. SGA also achieves high fitness by using the same solution as the previous environment. Therefore, GRN recalls memories stably, while the SGA temporarily falls into a local solution, which causes a small difference.

Overall, we can verify that GRN is more adaptive than SGA in the same environment. GRN can improve service stability as long as selecting the adaptable solution in other bandwidth to store in GRN.

The N_2 is a trade-off between the ability to recall the memory and to search around the memory. This environment settings requires phenotypic mutations around the optimal solution. Thus, after recalling the memorized solution, GRN, where $N_2 = 85$, has better fitness because many individuals are generated by the phenotypic mutation to search around.

TABLE V
RANDOMLY DEFINED REQUIREMENTS.

App. requirement for each session	Latency: T_s^{max}	Accuracy: A_s^{min} (mAP)
D1	0.14406 s	57.585 %
D2	0.06332 s	23.988 %
D3	0.07354 s	54.967 %
D4	0.12460 s	40.494 %

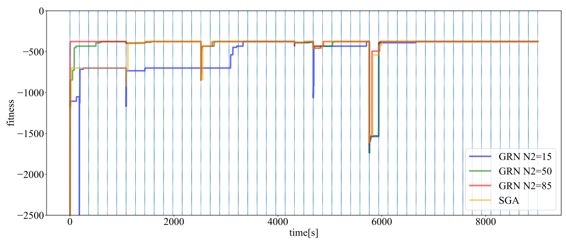


Fig. 5. The fitness of the proposed method (GRN) and SGA when the application requirement is in Table V with 50 patterns of bandwidth.

GRN, where $N_2 = 15$, recalled many memorized ones, so it has less searching ability. Dynamically setting N_2 would improve these adaptations.

D. Simulation of dynamic bandwidth with a new application

We set new application requirements and dynamic network bandwidth to evaluate adaptability in non-similar environments where the optimal solution structure is different from the one memorized by GRN. Variants in requirements tend to change the optimal solution structure more significantly than changes in network bandwidth. We evaluate the adaptability of the proposed method in cases where the solution structure changes to the not-trained ones and changes in some parts due to network variations. The application requirements are set as shown in Table V. In simulations, dynamic network bandwidth uses the same protocols as in Section IV-C1.

We describe the fitness transitions in Fig. 5. GRN, where $N_2 = 50$, shows higher adaptability than SGA in early stage, up to the seventh environment. It is considered that GRN captures certain features of optimization problems by memorizing multiple solutions, even if the solution structure differs from the stored ones. For example, particular bits of phenotypes of good solutions are often “0” in any case. In other words, GRN avoids solutions considered to be less adaptive. We need more evaluations of whether similar results can be obtained in other cases. After SGA found an adaptive solution, GRN showed slightly lower adaptability than SGA. SGA can search mainly around the discovered solution, while GRN searches around the memorized solution when the genotype mutation occurs.

Adaptation speed of GRN, where $N_2 = 15$, is slower. Since optimal solution are not memorized, it has to search for a new one by phenotypic mutation. However, GRN, where $N_2 = 15$, generates few individuals by phenotypic mutations and many ones by recalling other solution structure’s memory.

The proposed method consumes 392 W on average, while 421 W for SGA. Also, the unavailable time is 246 s for GRN and 257 s for SGA. GRN slightly reduces power consumption and improves stability due to the difference in early stage.

V. CONCLUSION

We proposed an optimization method to improve the adaptability of GA by GRN to distribute tasks for edge computing. We confirmed that our method could improve the adaptability to the switching of known environments by recalling the memorized solutions. Also in similar environments to the memorized cases, our method improved the adaptability. GRN escaped from local solutions by the multi-bit change of phenotype from single-bit genotypic mutation and the newly generated solutions by GRN. Our method is also effective in a non-similar environment because GRN captures certain features of optimization problems by storing multiple solutions. It avoided solutions that are considered to be less adaptive. Thus, the proposed method improved the adaptability of GA and achieved reductions in power consumption.

The matrix structure of GRN poses challenges in terms of scalability and computational complexity. The number of phenotypes that GRN can memorize depends on the mathematical model. It is necessary to confirm whether GRN can represent dynamic and complex systems even with increased scale. We also evaluate in more complex environments, such as more sessions, and compare with other optimization methods. To improve adaptability, dynamical adjusting of parameters is also our future work.

REFERENCES

[1] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine, “Digital twin: Origin to future,” *Appl. Syst. Innov.*, vol. 4 (2), no. 36, pp. 1–19, May 2021.
[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. of IEEE Conf. Comput. Vis. and Pattern Recog. (CVPR)*, Jun. 2016, pp. 779–788.
[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
[4] V. K. Quy, A. Chehri, N. M. Quy, N. D. Han, and N. T. Ban, “Innovative trends in the 6g era: A comprehensive survey of architecture, applications, technologies, and challenges,” *IEEE Access*, vol. 11, pp. 39 824–39 844, 2023.
[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[6] C. Caiazza, S. Giordano, V. Luconi, and A. Vecchio, “Edge computing vs centralized cloud: Impact of communication latency on the energy consumption of LTE terminal nodes,” *Comput. Commun.*, vol. 194, pp. 213–225, Oct. 2022.
[7] D. Whitley, “A genetic algorithm tutorial,” *Statistics and Comput.*, vol. 4, pp. 65–85, Oct. 1998.
[8] H. Shimonishi, M. Murata, G. Hasegawa, and N. Techasarntikul, “Energy optimization of distributed video processing system using genetic algorithm with bayesian attractor model,” in *Proc. of IEEE 9th Int. Conf. Netw. Softwarization*, Jun. 2023, pp. 35–43.
[9] P. A. Dnyane, S. S. Puntambekar, and C. J. Gadgil, “Method for identification of sensitive nodes in boolean models of biological networks,” *IET Syst. Biol.*, vol. 12, no. 1, pp. 1–6, Feb. 2018.
[10] R. A. Watson, G. P. Wagner, M. Pavlicev, D. M. Weinreich, and R. Mills, “The evolution of phenotypic correlations and “developmental memory”,” *Evolution*, vol. 68, no. 4, pp. 1124–1138, Dec. 2014.
[11] A. Wagner, “Does evolutionary plasticity evolve?” *Evolution*, vol. 50, no. 3, pp. 1008–1023, Jun. 1996.
[12] Online:https://github.com/AlexeyAB/darknet. Accessed 8 August 2023.