Energy Optimization of Distributed Video Processing System using Genetic Algorithm with Bayesian Attractor Model

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

Masayuki Murata
Osaka University
Osaka, Japan
murata@ist.osaka-u.ac.jp

Go Hasegawa
Tohoku University
Miyagi, Japan
hasegawa@riec.tohoku.ac.jp

Nattaon Techasarntikul
Osaka University
Osaka, Japan
techa.nattaon.cmc@osaka-u.ac.jp

***Abstract*—For the future cyber-physical system (CPS) society, it is necessary to construct digital twins (DTs) of a real world in real time using a lot of cameras and sensors. Video analysis for this purpose is expected to consume a large amount of traffic and power when all distributed data are gathered to the centralized data center. Hence, the energy efficiency of both networks and computers is a major challenge for the full-scale spread of CPSs and DTs. Toward this goal, we first propose a model to arbitrarily split and distribute the video analysis task to terminals, edge servers, and cloud servers and dynamically assign appropriate CNN models to them. System-wide optimization of such distributed processing can reduce overall system power consumption by reducing network bandwidth and efficiently utilizing distributed CPU/GPU resources. To realize this optimization in a real system, we also propose a model to estimate the GPU load, processing time, and power consumption of these devices based on massive experimental measurements. However, such a large-scale optimization is difficult because of the dynamic and multi-objective nature of the problem. We also found that due to the complexity and diversity of the real system, there are a large number of local optimum solutions in the solution space and it is difficult to find a global optimum solution. To solve the problems, we propose a new optimization algorithm composed of Genetic Algorithm and Bayesian Attractor Model. Finally, simulation evaluations are performed to demonstrate that the proposed method can minimize system power consumption and satisfy latency and recognition accuracy requirements of each video analysis, even under changing environmental conditions.**

***Index Terms*—Digital twin, video analysis, system optimization, energy efficiency, genetic algorithm, Bayesian Attractor Model**

I. Introduction

By constructing a cyber-physical system (CPS), for example, a digital twin (DT) of a warehouse, factory, mall, airport or even an entire city, real-time AI analysis of environmental data is expected to be increasingly used. The DT is constructed by collecting a large amount of data observed by sensor devices such as cameras and LiDARs via a network to a data center. Data processing requires a large amount of CPU/GPU resources to make decisions about recognition, identification, analysis, and actuation in the real world. In this way, the volume of traffic to be processed is expected to increase rapidly toward 2030 [1], especially for new applications in the era of 6G [2]. As a result, electricity consumption for both data transfer and processing is expected to increase at a very high rate in the near future [3]. Hence, the energy efficiency of the entire system is a major challenge for the full-scale spread of CPSs and DTs.

For DT applications, latency minimization is very important and, therefore, distributed processing is needed to take advantage of computing at terminals and edges, rather than collecting all data in the cloud [4]. Such distributed processing can reduce overall system power consumption by reducing network bandwidth and efficiently utilizing distributed CPU/GPU resources. However, this is not a trivial reallocation process from a cloud to edges or terminals; since it is not obvious when, where, and how much to reallocate.

To maximize distributed processing efficiency, it is essential to optimize both network resources and computing resources in an integrated manner. Although various studies have been conducted [4]–[7], few of them further integrate application processing optimization, such as joint optimization with video analysis algorithms. In distributed video analysis, processing time and analysis accuracy depend not only on the status of network and computing resources but also on the selection of video analysis algorithms. For instance, different CNN models have a trade-off between recognition accuracy and computing cost, which also affect processing time, and the distribution of the CNN processing task potentially has a trade-off between computing resource and network resource utilization. More importantly, these trade-offs are mutually interrelated. Therefore, we can expect further optimization of performance and power consumption of the distributed system, by totally integrating these factors.

In this paper, our challenge aims at integrated optimization, in particular, to minimize the power consumption of the entire system while satisfying constraints such as recognition accuracy and processing delay of each video analysis. In the next section, we describe the formal definition of the system architecture and mathematical model and then propose component modeling to estimate the GPU load, processing time, and power consumption of these devices based on massive experimental measurements and regression analysis. We also present the proposed optimization method composed of Genetic Algorithm (GA) [8] and Bayesian Attractor Model (BAM) [9], followed by performance evaluation and conclusion.

II. Distributed Video Analysis

When targeting the distributed video analysis described in the previous section, there are various possible approaches to decompose the video analysis. The most simple approach is to reallocate CNN processing from the cloud to the edge or terminal. However, in this case, it is difficult to arbitrarily distribute the processing load, as specific video sessions are processed entirely at the terminal, edge, or cloud. CNN processing cost is fairly large, and the edge or terminal has a relatively smaller computational capacity, so it is sometimes difficult to place CNN processing on the edge or terminal.

The more flexible allocation is to split and distribute the CNN pipeline, for example, the first half of the CNN layer to the edge and the last half to the cloud [10]. This approach may increase data traffic between the edge and the cloud, hence, the authors have proposed determining the best split point to minimize the traffic between them. A similar but more coarse approach is also used to split the video processing pipeline between edge and cloud servers [4]. Since each pipeline component is an independent processing unit, it is easy to split the load over the network. The authors have proposed an optimization scheme that takes into account the characteristics of each component of distributed applications. However, these approaches proposed a way towards splitting where the size of each split unit cannot be arbitrarily determined, and thus fine-grained computation task distribution is still a challenge.

Another approach is the division of the processing load in the time sequence; that is, video streams are split into frames and the frames are distributed to the terminal, edge, or cloud. As shown in Fig. 1, a video image captured by a camera is analyzed at the terminal for a certain portion of the frames, then the results of the analysis and the remaining unprocessed frames are sent to the edge. The same process is performed at the edge, and the results of the analysis at the terminal and the edge, as well as the remaining frames that have not been processed by either of them, are sent to the cloud. Finally, the cloud analyzes the remaining frames necessary to achieve the required video analysis accuracy. The results analyzed at these three locations are then integrated to produce the final analysis results. This kind of distributed processing enables the distribution of processing in any ratio on various devices, and thus we can optimize the entire system.

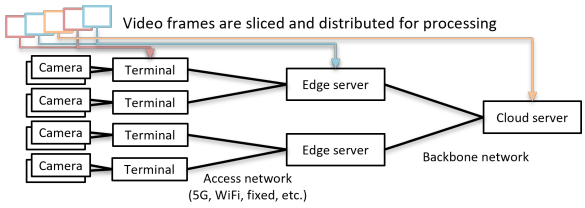

Fig. 1. Distributed Video Analysis Model

III. System optimization Strategy

In this study, we target the above-mentioned distributed video analysis system and investigate dynamic joint optimization of network, computing, and application (distributed video recognition via CNN). Many algorithms have been proposed to solve various optimization problems. For example, a virtual network assignment [11] is proposed using integer programming that incorporates the Markov Decision Process. Deep Reinforcement Learning is combined for long-term total cost optimization [12], and the deep neural network is used to optimize the number of resources allocated to the radio access network [13].

However, the difficulty in solving the optimization problem of distributed video analysis is its highly heterogeneous system and highly changing environment. For example, a network may be constructed of wired or wireless links that have very diverse characteristics. In particular, the performance of the wireless network is highly fluctuated and difficult to predict. The computational resources available on devices (terminal, edge server, or cloud server) are also varied, especially if they are equipped with different generations of GPUs. Furthermore, from an application point of view, the optimal strategy for each task is different because the maximum delay and recognition accuracy required for each type of task, such as person identification or vehicle collision avoidance, are different.

Therefore, we focus on GA [14], [15] to solve this multi-objective dynamic problem. The challenges in using GA are the convergence time and the local optimization problem. The former is that convergence to a new optimal solution takes time, especially when large environmental changes occur. The latter is that since the system modeled in this research is complex and heterogeneous, it could happen that many local optimum solutions exist and that global optimal solution cannot be reached no matter how much time is spent in evolution. To solve these problems, many advanced algorithms have been proposed to increase the diversity of the population [16], [17], however, they still struggle with the problem of convergence time.

In this study, we investigate the evolution of populations by introducing various external genes in order to react to large environmental changes. We assume that environmental changes occur periodically, and thus we retain some optimal genes obtained in the past experience. When the current environment is similar to the one in the past, the genes that have been retained are introduced into the population. Therefore, the proposed method needs to continuously monitor the current environment status, which is a high-dimensional information that contains network status, device status, session information and system topology. The inference problem of identifying the current environment status with the retained past status is very challenging due to high-dimensional and noise in the input information. For example, the monitored and estimated bandwidth available at wireless access links is not only inaccurate but fluctuating all the time. To this end, we propose using the BAM, which is known as one of the cognitive models of the human brain, to continuously identify the states of the system from noisy observations.

IV. System Model

A. System Architecture

Our proposed system model is shown in Fig. 2. Each video stream from the camera is split into frames. The frames are then distributed to the terminal, edge, or cloud to meet its requirements of end-to-end processing time (the sum of the processing delay at the devices and the transmission delay at the networks) and the recognition accuracy of the video analysis (the weighted average of the recognition accuracies at the devices by their respective processing portions). The optimizer takes this information as constraints and calculates system control information consisting of processing portion at each device and its CNN model applied to minimize power consumption of the entire system (devices and networks) yet satisfy the constraints. The reason is that the available resources on networks and servers are changing over time, and thus the optimizer continuously monitors these environmental changes and dynamically recalculates the system control information.

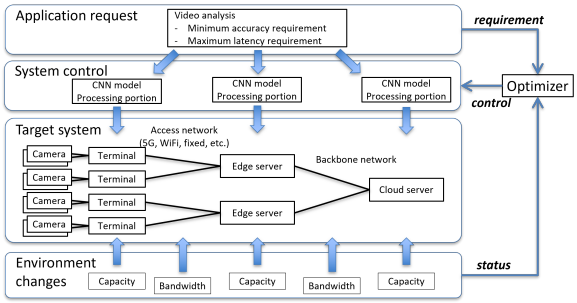

Fig. 2. System Model.

B. Mathematical Model

We define the optimization of this distributed video processing system as a combinatorial optimization problem, which is expressed in the following three equations: (1) constrain the end-to-end processing time per frame, (2) constrain the average video analysis accuracy, and (3) minimize the overall power consumption of the system. In this study, the optimization problem is to find the best combinations of processing portions and CNN model selections that satisfy both equations (1) and (2) and minimize equation (3).

The first equation is the processing time constraint, as follows:

$$T_s = \sum_{d \in D^s} T_s^d(t) + \sum_{n \in N^s} T_s^n(t) \leq T_s^{max}, \forall s \in S \quad (1)$$

where S is the set of all video sessions, D^s is the set of devices used by session s. N^s is the set of networks (access/backbone) used by the session s. T_s is the end-to-end processing time of session s and T_s^{max} is given as its maximum constraint. $T_s^d(t)$ is the video analysis time for one frame on device d of session s at time t, and $T_s^n(t)$ is similarly the transmission delay for one frame on network n, which is defined in the next section.

The second equation is the accuracy constraint as follows:

$$A_s = \frac{\Sigma_{d \in D^s} A^{M_s^s(t)} W_s^d(t)}{|D^s|} \geq A_s^{min}, \forall s \in S \quad (2)$$

where A_s is the video recognition accuracy of session s and A_s^{min} is given as its minimum constraint. $M_s^d(t)$ is the CNN model selected by session s on device d. and its accuracy is given by the value of the mean average precision (mAP) of the recognition accuracy. Let $W_s^d(t)$ be the portion of video frames from session s and processed on device d. Note that we let $W_s^d(t) = 0$ if $W_s^d(t) < 0.1$ to avoid inefficient allocation and encourage the less loaded device to turn off the power. $|D^s|$ is the number of devices that session s is using, which is 3 for our system model shown in Fig. 2.

Finally, the total system power consumption P is the sum of the power consumption of all devices D_s and networks N_s used by all sessions. Let $P^d(t)$ be the power consumption of device d and $P^n(t)$ be the power consumption of network n.

$$P = \sum_{d \in D^s} P^d(t) + \sum_{n \in N^s} P^n(t) \quad (3)$$

In the next section, equations (1) to (3) are used as functions of the processing portion $W_s^d(t)$ and the selection of the CNN model $M_s^d(t)$.

V. Component Modeling

A. Device model

Fig. 3 shows photos of our experimental system for testing digital twin applications. The devices, including terminals, edge servers, and cloud servers, are tested in this section. Cameras are installed at the corners of the ceiling, and the field contains robots and objects.

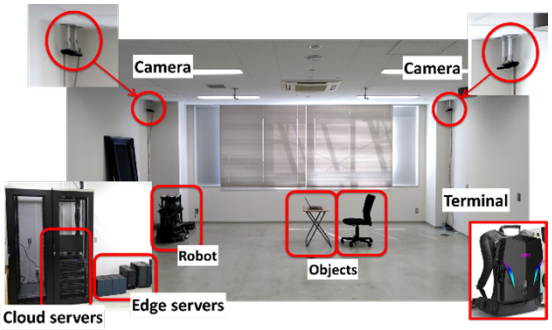

Fig. 3. Tested Devices.

As part of the mathematical model described above, we modeled the devices by performing massive experimental measurements using three types of devices, IA1, IA2 and IA3 shown in Table I. The devices IA1 and IA2 are used as a terminal or an edge server, whereas IA3 is used for the cloud server. Explanations on the specifications are given later in this section.

We used the Yolo-v3 implementation [18] for video analysis. We have downloaded pre-trained CNN models from the same site. The mAP and the number of floating operations per

TABLE I
DEVICE SPECIFICATION

Device	IA1	IA2	IA3
CPU	Core i7 8700T	Core i9 10940X	Xeon GOLD 6226R x2
CPU TDP	35W	165W	150W x2
GPU	Nvidia GTX 1070	Nvidia RTX A5000	Nvidia Tesla T4
GPU (FP32)	6.463 Tflops	27.77 Tflops	8.141 Tflops
GPU TDP	150W	230W	70W

frame of the models are also obtained from the same site using MS-COCO dataset, and we used the same values as listed in Table II. The video [19] from YouTube Bounding Boxes' Object Detection in Video Segments-validation set [20] was used as the evaluation video and correct answer labels in this experiment.

Using the devices described above, we varied the number of video sessions and the processing portion in various ways and measured GPU utilization and power consumption of each device, the processing time per frame, and mAP. Then, we have developed models to estimate these values.

TABLE II
CNN MODELS

Model	mAP	Floating operations
Yolov3-tiny	33.1%	5.6B / frame
Yolov3	55.3%	65.9B / frame
Yolov3-spp	60.6%	141.5B / frame

1) GPU load model: We first propose a model to estimate the GPU load of a device (one of the devices in Table I). Based on the analysis of the measured data, we found that the GPU load, obtained from nvidia-smi command, is almost proportional to the estimated floating point operations per second (FLOPS) on the device, and thus we build a model to estimate the GPU load $L_d(t)$ of the device d as follows:

$$L^d(t) = \frac{\sum_{s \in S^d} O_s^d\ (t)}{C^d E^d} \tag{4}$$

where $O_s^d(t)$ is the FLOPS estimate of session s, and S^d is the set of sessions on the device d. C_d is the performance of the GPU for the calculation of floating point (FP) 32 on the device d, which is shown in Table I, and E^d is the estimated FP32 calculation efficiency of the GPU on the device d determined by regression analysis of the measured data and shown in Table III below. We also found that the model fits the experimental results if we assume that the FLOPS estimation of a video session consists of two parts; CNN processing and fixed processing. CNN processing part is the sum of the model-dependent part and the model-independent part, and its FLOPS is proportional to the number of frames processed. On the other hand, the fixed processing part is the overhead processing other than CNN, such as decoding/encoding, and its FLOPS is proportional to the total number of frames received by the device. Therefore, $O_s^d(t)$ is modeled as follows:

$$O_s^d(t) = ((O^{M_s^d(t)} + O^A)W_s^d(t) + O^B)FPS \tag{5}$$

where, $O^{M_s^d(t)}$ is the number of floating operations (FLOPs) of the model-dependent part of CNN processing shown in Table II selected by session s on device d, and O^A is the model-independent part of CNN processing and is shown as *Floating operation A* in Table III. For example, in the case of Yolo-v3, they are 65.9B:1.5B per frame, and thus the model-dependent part is the majority. O^A is the FLOPs of the fixed processing part and is shown as *Floating operation B* in Table III. This part is much smaller than CNN processing, as expected, but is also needed for accurate GPU load estimation. FPS is the number of frames per second, and the sample video used in this paper is encoded at 30 fps. These parameter values were obtained by regression analysis based on the measured data. Fig. 4 shows the comparison between the measured GPU load and the estimated GPU load for these three devices. The estimation error increases slightly in the very high load situation, but is still within 10%. The Rooted Mean Square Error (RMSE) of the estimation was 4.9%.

TABLE III
GPU MODEL DATA

	IA1	IA2	IA3
GPU efficiency (E^d)	0.4	0.48	0.39
Floating operations A (O^A) per frame	1.5B	8.1B	2.2B
Floating operations B (O^B) per frame	0.1B	0.4B	0.1B

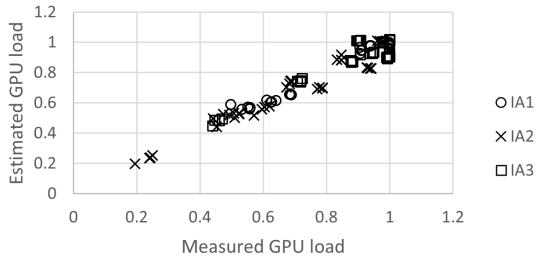

Fig. 4. GPU load estimation accuracy

2) Processing time model: Next, we propose a model to estimate the processing time of a frame. Based on measurement data analysis and the above GPU model, we build the processing time model by the total FLOPs of CNN processing, including model-dependent and model-independent parts, and fixed processing divided by GPU capability and load. Thus, we obtained an estimation equation for the processing time of a frame, $T_s^d(t)$, as follows:

$$T_s^d(t) = \frac{O^{M_s^d(t)} + O^A + O^B(t)}{C^d E^d L^d(t)} \tag{6}$$

Fig. 5 shows the comparison between the measured and estimated processing times for three devices. The RMSE of the estimation was only 1.7 msec, which was very accurate.

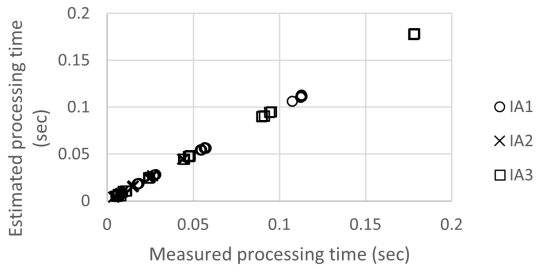

Fig. 5. Processing time estimation accuracy

3) Power consumption model: Finally, we propose a model to estimate the power consumption of a device. We assume that power consumption is the sum of CPU power consumption, GPU power consumption, and idle power, and we propose a formula for the power consumption $P^d(t)$ of the device d as follows:

$$P^d(t) = \begin{cases} P^d_{IDLE} + P^d_{CPU-TDP}\alpha^p \\ +P^d_{GPU-TDP}L^d(t)\beta^p & \text{if } L^d(t) > 0 \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where $P^d_{CPU-TDP}$ and $P^d_{GPU-TDP}$ are Thermal Design Power (TDP), i.e. a maximum power consumption, of the CPU/GPU, which we obtained from their catalog as shown in Table I. P^d_{IDLE} is measured as a minimum power consumption of a device when we run the device without running any processes other than OS. The idle consumption for IA1, IA2 and IA3 was 21.3W, 98.3W, and 247.3W, respectively. Since it was difficult to obtain a clear estimate of CPU load through the measurement data because other processes are running on the same device, we used a common value for all cases. Then, the values of (α^p, β^p) were determined by regression analysis as (0.97, 0.78), (0.97, 0.67) and (0.29, 0.81) for IA1, IA2 and IA3, respectively. We also assume that if there is no load on the device, it should be powered off, and thus the power consumption is zero.

Fig. 6 shows the comparison between the measured and estimated power consumption for three devices. The actual power consumption of a device was measured using a watt meter connected to the AC input of the device. Unfortunately, the power estimation is not very accurate due to various reasons, including the difficulty in estimating CPU load and power consumption by cooling fans and others. As a result, the RMSE was 24.2W, and an accurate estimate of power consumption is a future issue.

B. Network model

Transmission delay and power consumption in a network node are not measured in this paper as we consider them to be part of the public infrastructure. The access network would be cellular networks and the core network would be a large-scale

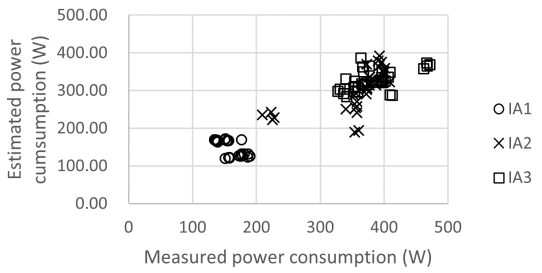

Fig. 6. Accuracy of Power Consumption Estimation

fiber optic network, which is shared by many users. Therefore, we assume that the available bandwidth on the network cannot be predetermined and is dynamically measured in real time when the video stream uses the network. We also assume that the power consumption for this task is only proportional to the amount of data sent over the network. Thus, we modeled the computational model for network latency $T^n_s(t)$ and power consumption $P^n(t)$ as follows:

$$T^n_s(t) = M\Big(1 - \sum_{d \in D^{s,n}_{pass}} W^d_s(t)\Big)/B^n(t) \quad (8)$$

$$P^n(t) = \sum_{s \in S} M\Big(1 - \sum_{d \in D^{s,n}_{pass}} W^d_s(t)\Big)p^n \quad (9)$$

where M is the number of bits per frame, $B^n(t)$ is the bandwidth available at time t on the network n, and p^n is the power consumption for one bit transmission on the network n. In the following evaluation, M is set to 470Kbit, p^n is 193×10^{-9} and 60×10^{-9} for the access and core networks, respectively. These values are calculated from catalog data for carrier-grade network equipment. The available bandwidth $B^n(t)$ is given in the simulation. In the above equation, $D^{s,n}_{pass}$ is the set of devices that session s has passed before reaching node n. This means that a video frame processed by a device will not consume network bandwidth after the device.

VI. Optimization Method

Our proposed optimization method shown in Fig. 7 uses a GA to dynamically search for genes as optimal solutions in changing environments. To improve initial convergence and reaction to large environmental change, BAM monitors the environment and when it finds the current environment to be close to one of the retained environments, the corresponding retained genes are introduced into the population. In the following, we will first explain the basic solution search using GA, and then introduce external genes by BAM.

A. Dynamic solution search by GA

In the proposed method, the states of the entire system are expressed as a gene in an individual. A population is a set of individuals who have various genes. An individual that has the highest fitness for the current environment, where the constraints of all sessions are met and the total power consumption

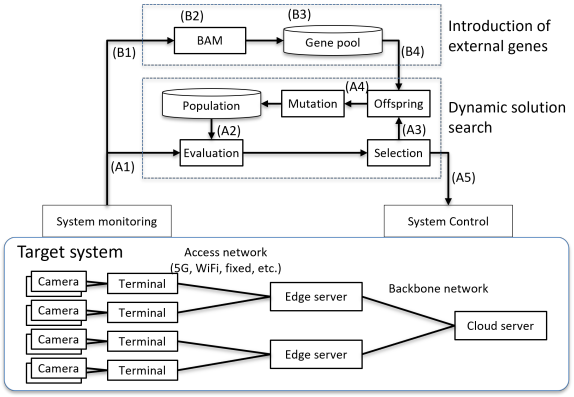

Fig. 7. System Model.

is minimum, is selected as the best individual, and the entire system is controlled by its gene. The genes express the system control parameters, portion of the processing, and selection of the model at terminal, edge and cloud for all sessions; thus, it is defined as a set of these values:

$$[W_s^d(t)\ for\ \forall s \in S\ and\ \forall d \in D^s,$$
$$M_s^d(t)\ for\ \forall s \in S\ and\ \forall d \in D^s].$$

Then, the fitness value F of an individual with such a gene is obtained using equations (1)-(3) and their associated equations, as follows:

$$F = -P + \sum_{s \in S} min(T_s^{max} - T_s, 0)\alpha^f + \sum_{s \in S} min(A_s - A_s^{min}, 0)\beta^f. \quad (10)$$

The lower the power consumption P, the higher the fitness F, which takes negative values. If the processing time exceeds its limits or the recognition accuracy is lower than its limits, the fitness F is reduced accordingly. α^f and β^f are hyper-parameters for constraints and were set to very large values because we place more importance on the fulfillment of the constraints than on power consumption. We also evaluate a method without considering power consumption, which uses the following equation as a fitness function.

$$F' = \sum_{s \in S} min(T_s^{max} - T_s, 0)\alpha^f + \sum_{s \in S} min(A_s - A_s^{min}, 0)\beta^f. \quad (10')$$

In GA optimization, the entire system is dynamically optimized by repeating steps (A1) through (A5) below, as shown in Fig. 7.

(A1) Measure the state of the networks and devices, and give this as the environmental condition.

(A2) Decode the genes of each individual and update its fitness value under the environmental condition.

(A3) Select elite individuals with higher fitness values from the population and form a new population consisting of them and their offspring.

(A4) Produce offspring from elite individuals and mutate its some genes.

(A5) Update the processing portion and CNN model selection in the devices for each video session, based on the information about the gene of the individual that has the highest fitness value.

B. Introduction of external genes by BAM

We also propose the evolution of populations by introducing various external genes to react to large environmental changes.

1) Environmental state inference: The proposed method continuously monitors the current environment status, and BAM takes a time series of noisy and high-dimensional input data, *Observation*, and gradually updates its belief about which retained set of status, *Attractors*, is most similar to the current input data via Bayesian inference. The mechanism of BAM is not described in this paper, but it is introduced in the literature [9], and we will describe how it is applied to the optimization problem.

In this paper, we assume that the distributed video analysis application is assigned with dedicated terminals, edge servers and cloud servers, and thus we assume that the number of devices and its available capacity are unchanged. Instead, access and backbone networks are shared with many other users, and their wireless capacity fluctuates a lot. Therefore, we define that the environmental state includes the available monitored bandwidth $B^n(t)$ of the network link n $(0 \leq n \leq N)$ at time t as follows:

$$\text{Observation: } [B^0(t), B^1(t), \ldots, B^N(t)].$$

Then we also set a number of attractors, which are one of the past experienced environmental states, as follows.

$$\text{Attractors: } [A_i^0, A_i^1, \ldots, A_i^N], 1 \leq i \leq |A|,$$

where A_i^n is the i-th attractor and its dimension is N, which is the number of monitored network links. $|A|$ is the number of attractors. In our system, inference is made in the following steps, which is also shown in Fig. 7:

(B1) Observe the available network bandwidth as system status

(B2) Input the observation into BAM and let it change its decision state to determine attractor(s)

If multiple attractors are found to be similar to the current status, then all of them are sent to the next steps with its strength of belief.

2) Introduction of genes via two-point crossover: In this paper, we assume that pairs of an attractor and a gene are given in advance. The attractor corresponds to the past experiences of the environment, and the gene is the one of an individual that has the highest fitness in that environment. We call the set of pairs a gene pool. When the BAM identifies that the current status is similar, not exactly the same, to the past status, its associated gene is introduced to the current population as follows. If multiple attractors are selected, the genes of these attractors are randomly introduced into the population in proportion to the strength of their beliefs.

(B3) Take the gene(s) associated with the selected attractor(s) from the gene pool.

(B4) The gene is introduced into the offspring of an individual.

However, in fact, the gene is the best from past experience and not necessarily the best from the current situation, and thus an individual with the gene may be weeded out very soon, or the population with the gene may experience catastrophic collapse. Therefore, we propose to gradually introduce the gene via two-point crossover, that is, two random points are chosen on the individual gene strings, and the genes are exchanged at these points. This means that system control parameters are not changed entirely, but part of it, i.e., several sessions, is randomly chosen and replaced, and then the gene is tested if it works well or not.

VII. Optimization Results

Some performance evaluation results are shown in this section based on computer simulations with the actual component modeling results described in the previous sections.

A. Evaluation settings

The proposed method is evaluated with the system model shown in Fig. 2. The parameters used in the simulation experiments are shown in TABLE IV. The latency requirement, accuracy requirement, access network bandwidth, and backbone network bandwidth are randomly chosen from the range shown in the table. Network topology, i.e., connections between terminal and edge servers, and connections between edge servers and cloud servers, is chosen at random. Device allocation, i.e., devices used by a video session, is also randomly chosen.

TABLE IV
Simulation Parameters

Number of video sessions	20
Video bitrate	14.1 Mbps
Video frame per second	30 FPS
Latency requirement	50 msec – 200 msec
Accuracy requirement	mAP 0.2 - 0.7
CNN models	Yolo-tiny / Yolo / Yolo-spp
Number of terminals	10 (IA1)
Number of edge servers	4 (IA2 or IA3)
Number of cloud servers	2 (IA2 or IA3)
Access network bandwidth	10 Mbps – 50 Mbps
Access network propagation delay	10 msec
Backbone network bandwidth	10 Mbps – 500 Mbps
Backbone network propagation delay	10 msec
Population size	500
Mutation rate	0.2
Attractors for external genes	5 / 10 / 20

B. Simple Genetic Algorithm (Simple GA)

First, we evaluate the traditional simple GA algorithm as a baseline. In Fig. 8, the traces of the fitness value of the best individual in a population are shown. Multiple plots are shown for different simulation runs with exactly the same settings. As shown in this figure, we found that the evolution has two modes; first the fitness value is quickly improved and reaches some initial convergence point, then the fitness value improves slowly and occasionally. We also found that, although the fitness values are very similar in the different runs, their control parameters, or gene, can be quite different. This means that there are many local optimum solutions that have almost the same goodness in the solution space. Therefore, the local optimum can be found easily, and it takes time to jump to other better local optimums to approach the global optimum.

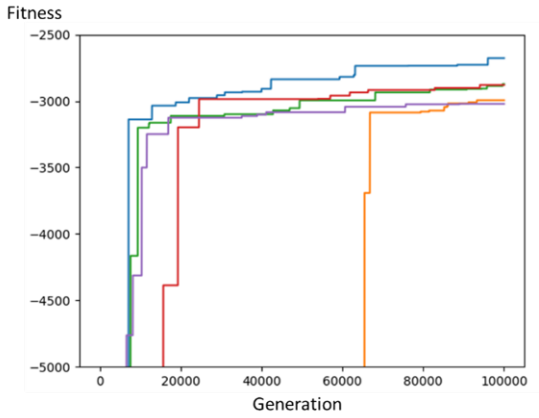

Fig. 8. Fitness of 5 runs with the same configuration (Simple GA).

In Fig. 9, the total power consumption of the system, the latency violation ratio, and the accuracy violation ratio of a run are plotted as an example. We can confirm the characteristic behavior of the proposed optimization algorithm. In the first phase, two violation ratios decrease first, since the violation has a much higher penalty than a higher power consumption from an application point of view, as shown in Eq. (10). In this phase, power consumption increases and decreases randomly because the optimization algorithm focuses on solving the violations. In the second phase, around the 9,000th generation, both constraints of all sessions have been met, and then the power consumption starts to continuously decrease. This is an intended algorithm behavior, but we confirm that the Simple GA algorithm takes time for initial convergence.

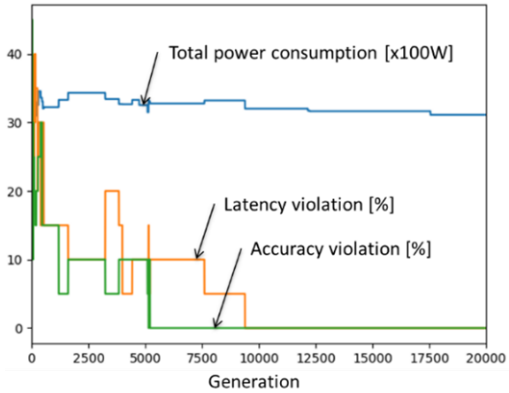

Fig. 9. Total power consumption and violations (Simple GA).

C. GA with BAM

We compared the algorithm with and without external gene injection using BAM. In Fig. 10, the fitness averaged over 5 runs with the same configuration is plotted for Simple GA and GA with BAM having 5, 10, and 20 attractors. The result of GA is the same as in Fig. 8, and 5 runs in that figure are averaged to one in this figure; thus we confirm again that Simple GA takes time for initial convergence.

In the cases of GA with BAM, the attractors and their associated gene were generated in advance in randomly generated 5, 10, and 20 environments. The result shows that although the current environment is not exactly the same as the retained environment, the algorithm can quickly find better solutions using the fraction of retained genes. As the number of attractors increases, the algorithm finds better solutions. We also found that no matter how long the population evolves, these small differences do not dissolve. This would mean that there are still many local optimums close to the global optimum and that it is very difficult to reach the final global optimum no matter how close the algorithm approaches it.

D. Behavior in dynamic environment

In Fig. 11, Simple GA and GA with BAM are compared in a dynamic environment, where network bandwidths are randomly changed within the range shown in Table IV every 10,000 generations. We confirm that Simple GA takes time for initial convergence, but, as it experiences environment change many times, convergence after the change is improved. However, it is still stuck in a local optimum that is no better than GA with BAM. GA with BAM has good convergence and always finds better solutions than Simple GA, but in this case, it is sometimes stuck with a lower fitness value.

Lastly, we investigate the total power consumption of the system, which is composed of network nodes and processing devices. In Fig. 12, we tested Simple GA and GA with BAM, however, Eq. 10' was applied to Simple GA to investigate the ability to save power. We also compared the algorithm in both static environment, where no network bandwidth changes are applied, and dynamic environment, where network bandwidths are randomly changed at every 10,000 generations. Without considering the power consumption in Simple GA, the total power consumption in a static and dynamic environment does not have much difference, which is 3182W and 3223W, respectively. On the other hand, with the consideration of power consumption by GA with BAM, the total power consumption in the static environment can be reduced to 2643W, while it increases to 2750W in the dynamic environment. To compare these algorithms, GA with BAM can reduce power consumption by approximately 20% and 17%, respectively, in a static and dynamic environment, compared to Simple GA.

VIII. Conclusion

In this paper, we proposed a model to cooperatively optimize the entire system by distributing the video analysis processing using CNN (Yolo-v3) in each frame and optimizing

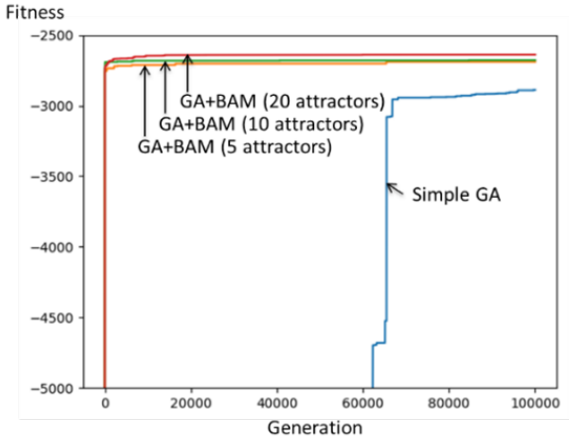

Fig. 10. Fitness on average of 5 runs

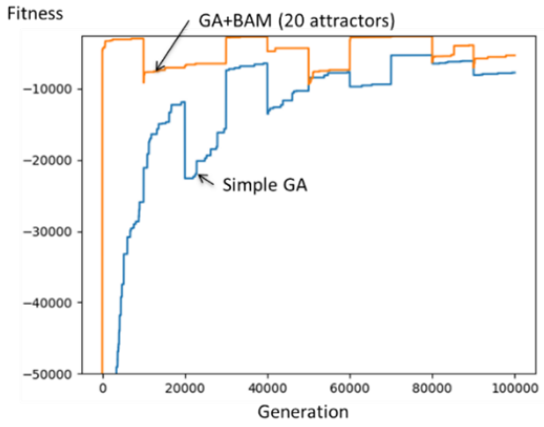

Fig. 11. Fitness of dynamic environment in average of 5 runs

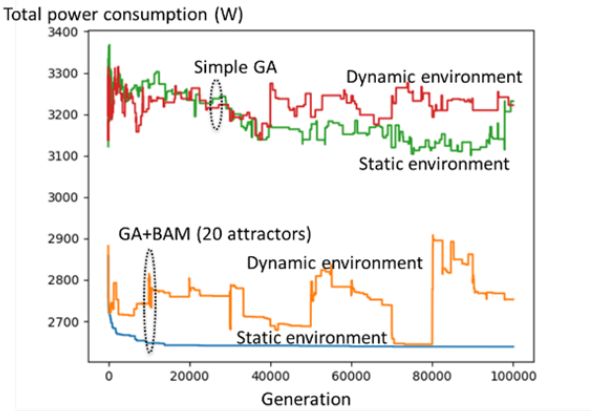

Fig. 12. Total power consumption of both static/dynamic environment on average of 5 runs

the parameters of each CNN at the same time. We then proposed optimization algorithms based on GA and BAM models to find the optimal solution that minimizes the power consumption of the entire system while simultaneously satisfying the latency and recognition accuracy requirements of various video sessions. Finally, simulation evaluations were conducted based on models measured on actual equipment, showing the effectiveness of the proposed method, especially its ability to adaptively derive the optimal solution under changing environmental conditions. In the future, we plan to further extend the proposed method to model higher-dimensional system conditions, such as the response to environmental changes other than network fluctuations.

ACKNOWLEDGMENT

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

REFERENCES

[1] A. S. Andrae, "Prediction studies of electricity use of global computing in 2030," *International Journal of Science and Engineering Investigations*, vol. 8, no. 86, pp. 27–33, 2019.

[2] R. Li *et al.*, "Towards a new internet for the year 2030 and beyond," in *Proc. 3rd Annu. ITU IMT-2020/5G Workshop Demo Day*, 2018, pp. 1–21.

[3] A. S. Andrae and T. Edler, "On global electricity usage of communication technology: trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.

[4] K. Rao, G. Coviello, W.-P. Hsiung, and S. Chakradhar, "Eco: Edge-cloud optimization of 5g applications," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2021, pp. 649–659.

[5] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.

[6] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 6, pp. 742–763, 2020.

[7] Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic task allocation and service migration in edge-cloud iot system based on deep reinforcement learning," *IEEE Internet of Things Journal*, 2022.

[8] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[9] S. Bitzer, J. Bruineberg, and S. J. Kiebel, "A bayesian attractor model for perceptual decision making," *PLoS computational biology*, vol. 11, no. 8, p. e1004442, 2015.

[10] R. Mehta and R. Shore, "Deepsplit: Dynamic splitting of collaborative edge-cloud convolutional neural networks," in *2020 International Conference on COMmunication Systems and NETworkS (COMSNETS)*, 2020, pp. 720–725.

[11] M. Shen, K. Xu, K. Yang, and H.-H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*. IEEE, 2014, pp. 61–70.

[12] F. Wei, G. Feng, Y. Sun, Y. Wang, and Y.-C. Liang, "Dynamic network slice reconfiguration by exploiting deep reinforcement learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[13] H. Chergui and C. Verikoukis, "Opex-limited 5g ran slicing: an over-dataset constrained deep learning approach," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[14] I. Pathak and D. P. Vidyarthi, "A model for virtual network embedding across multiple infrastructure providers using genetic algorithm," *Science China Information Sciences*, vol. 60, no. 4, pp. 1–12, 2017.

[15] G. Zhao, S. Qin, G. Feng, and Y. Sun, "Network slice selection in softwarization-based mobile networks," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 1, p. e3617, 2020.

[16] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 211–218.

[17] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[18] "Alexeyab/darknet: Yolov4 / scaled-yolov4 / yolo - neural networks for object detection," https://github.com/AlexeyAB/darknet, accessed: January 19, 2023.

[19] "Przeglad muzyczny 2011- by raz pewien pan," https://www.youtube.com/watch?v=AJbQP-rIwCY, accessed: January 19, 2023.

[20] "Youtube-bb dataset — google research," https://research.google.com/youtube-bb/download.html, accessed: January 19, 2023.