Hierarchical Bayesian Attractor Model for Dynamic Task Allocation in Edge-Cloud Computing

Tatsuya Otoshi, Masayuki Murata, Hideyuki Shimonishi, Tetsuya Shimokawa
Osaka University

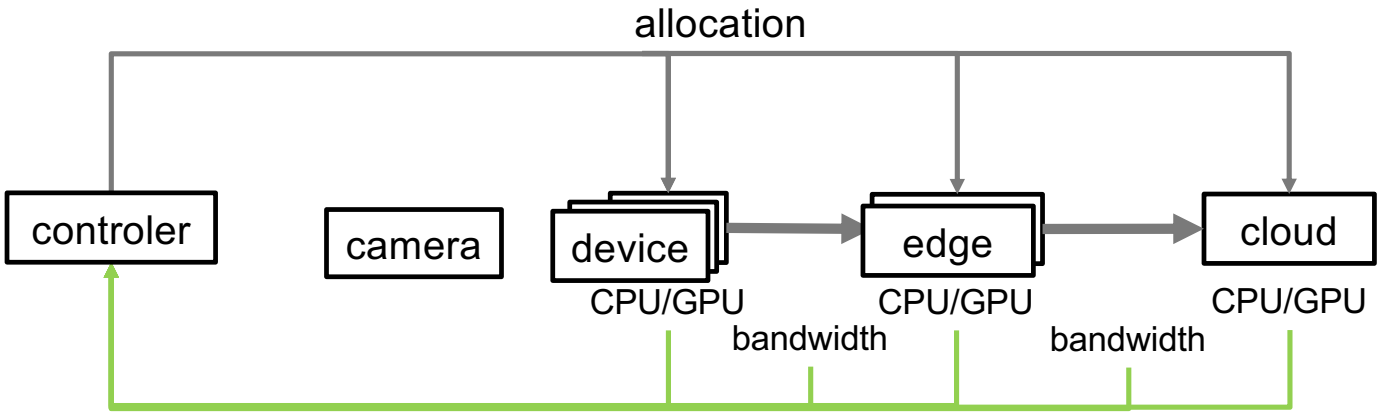
Edge AI Computing

- Edge Computing
  - Edge computing allows for information processing to take place close to the terminal device, rather than in the cloud.
  - This can reduce latency for latency-sensitive applications such as teleoperation.
- Edge AI
  - Large AI models may be difficult to run at the edge due to limited computational resources.
  - AI distillation techniques can be used to convert large models into smaller ones that can be run at the edge.
  - Application development environments like Tensorflow.js are being developed to support both server-side and client-side AI processing at the edge.

It is important to consider trade-offs when allocating processing to terminals, edges, and the cloud, including computational resources, accuracy, and power consumption.

Computer Vision of Video Streaming by Edge-Cloud Computing

- System
  - AI processing, such as image processing, is performed on terminal-sensed data like camera images.
  - The system can place restrictions on end-to-end processing delay and raw AI processing accuracy.
  - The system aims to satisfy user requirements while optimizing an objective function, such as power consumption.
- AI model
  - The size of the AI model used for processing depends on the distillation technique, with smaller models having lower accuracy but lower computational resource requirements.



Task Allocation in Quasi-Static Environment

- Cloud-edge task allocation problem involves deciding how much processing should be done at the terminal, edge, or cloud.
- The task allocation is dynamic and may require reallocation due to changing processing accuracy and latency requirements.
- The task allocation problem is a combinatorial optimization problem that is often treated as a static task allocation problem, using methods such as sequential task allocation or heuristics to find approximate solutions.
- These methods can lead to **deviations from the optimal solution**, especially in quasi-static environments where application-derived fluctuations have settled down.

It is important to maintain near-optimal settings in quasi-static situations while also being able to respond to changes in dynamic situations.

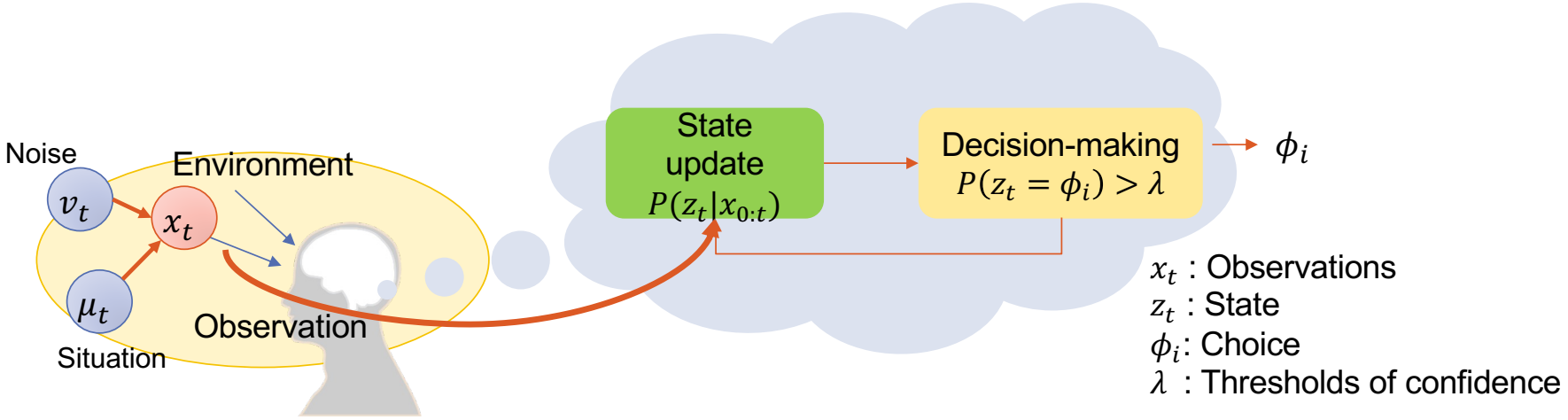
Approach

- Our group proposes a solution for dynamic assignments that combines **short-term and long-term** decision-making.
- Short-term decision-making
  - Short-term decisions are made quickly for dynamic changes, while long-term decisions compute optimal solutions for quasi-static environments in the background.
- Long-term decision-making
  - The long-term decision-making solution is used as a candidate solution for short-term decision-making, which selects a solution similar to the current situation in the past quasi-static environment.
  - Conventional static optimization can be used for long-term decisions because time-consuming computation is acceptable.

This paper focuses on short-term decisions with given solutions in a quasi-static environment, **extending the Bayesian attractor model** to make decisions that can withstand noise in real environments.

Bayesian Attractor Model

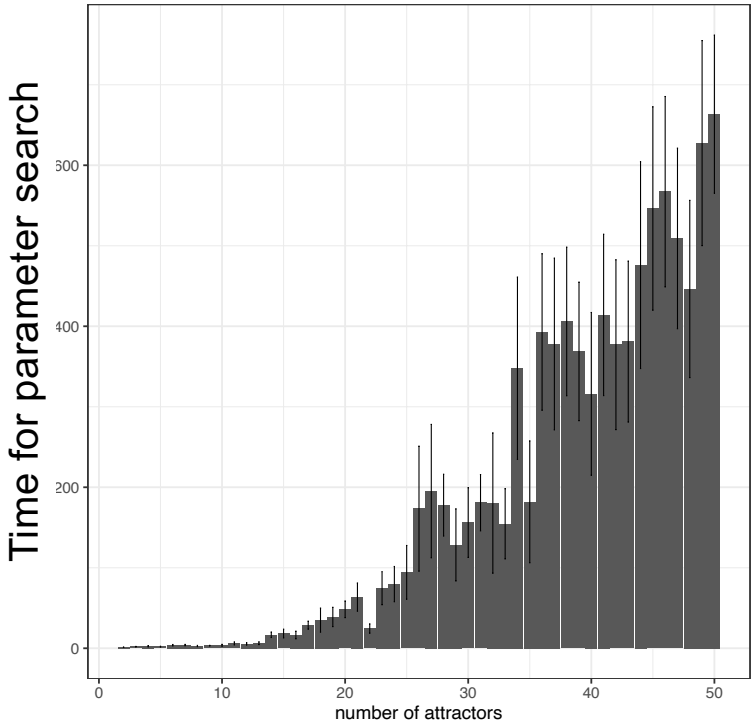
- The Bayesian attractor model (BAM) is a model of brain decision making that consists of observed values, internal states, and attractors.
- Decisions are made by updating the internal state based on the observed values and the internal state's association with a particular attractor.
- The model includes noise, allowing for revision and estimation of the internal state and predictions even when there are observation errors.



[1] S. Bitzer (number)J. Bruineberg, and S. J. Kiebel (singer), "A bayesian attractor model for perceptual decision making," *PLOS Computingal Biology*, vol. 11, no. 8, p. e1004442, Aug. 2015.

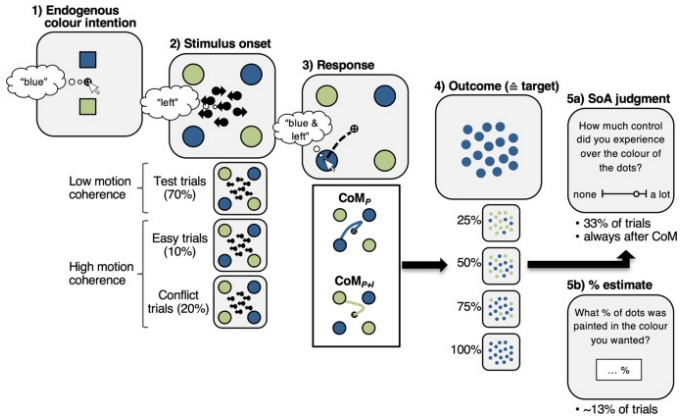
Limitation of Number of Attractors

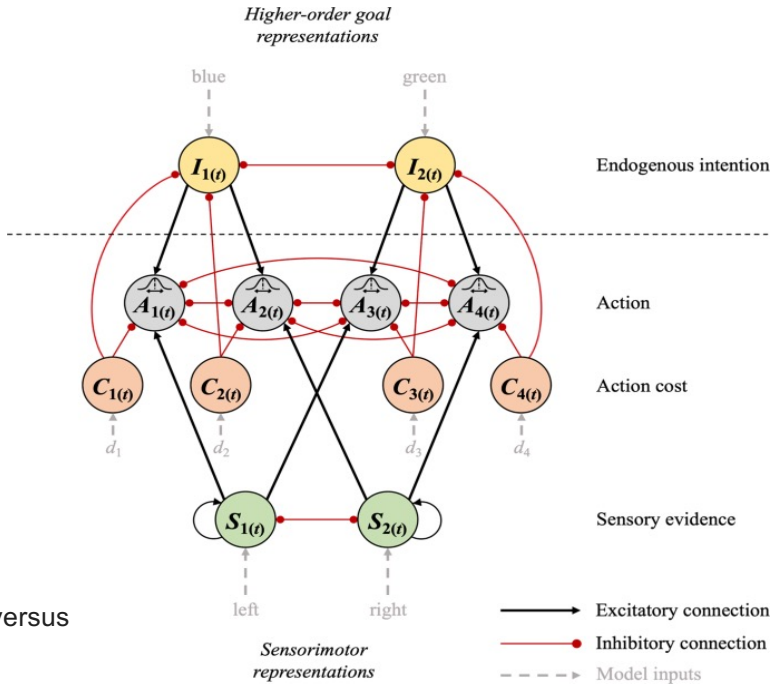
- Investigate the relationship between the number of attractors in an attractor set and the difficulty of parameter tuning
- Method
  1. Randomly generated parameter (sigmoid slopes) candidate sets
  2. Perform BAM classification on predefined attractors
  3. For attractors a and b with close representative values, give the observed values moving from a to b
  4. Iterate 1-2 candidates until the accuracy of the classification result exceeds the threshold (95%)
  5. Measure the number of parameters explored while changing the number of attractors
- Result
  - More severe parameter tuning is required as the number of attractors increased



Hierarchical Decision of Human

- Humans make fast decisions by hierarchically organizing their behavior, called chunking.
- Intention is a higher-level abstract action that combines multiple primitive actions and is also a hierarchization of time direction.
- Some models of hierarchical decision making include the excitation-inhibition relationship between nodes of a hierarchical neural network, the hierarchical Gaussian filter, and attention as an upper layer of decision making.

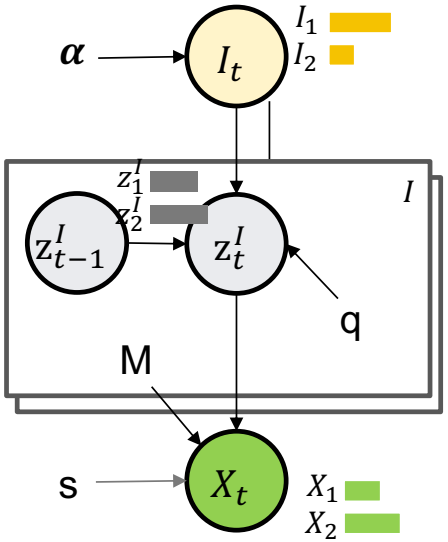




Löffler, Anne, et al. "A Hierarchical Attractor Network Model of perceptual versus intentional decision updates." *Nature communications* 12.1 (2021): 1-17.

Hierarchical BAM（HBAM)

- Recognizes situations based on bottom-up observational input and top-down intentions
  - Reflects differences in intentions as a mixed model
  - BAM attractor set switches according to intention → Narrow down the search space
  - Triggered by mismatch with observation, intention changes → shift of search space
- Advantages of a hierarchical structure
  - Reflects a hierarchical structure of time granularity (long-term at the top, short-term at the bottom)
  - Increases the total number of attractors while keeping the number of attractors per BAM constant
  - Structures the understanding of details from overview by mapping intentions to attractor sets and decisions to attractors



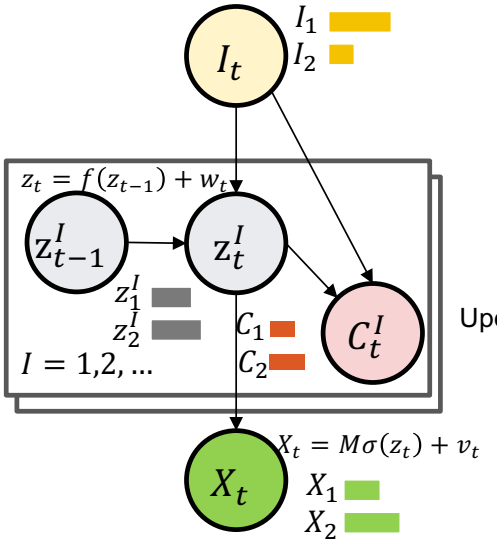
$$I_t = \begin{cases} I_t & (with\ 1-p) \\ I \sim Mult(\boldsymbol{\alpha}). & (with\ p) \end{cases}$$

$$z_t^{I_t} = f\left(z_{t-1}^{I_t}\right) + qw_t$$

$$X_t = M\sigma\left(z_t^{I_t}\right) + sv_t$$

State update by parallel and series update

- Parallel update
  - Simultaneous state updates for multiple intentions
  - Bayesian update of the entire model
    - Update multiple BAMs weighted by posterior probability of intention
    - Update posterior distribution of intentions according to the likelihood of each BAM



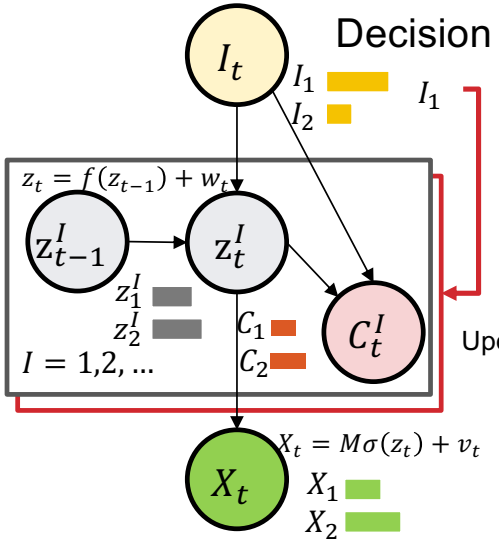
$$I_t = \begin{cases} I_t & (with\ 1-p) \\ I \sim Mult(\boldsymbol{\alpha}). & (with\ p) \end{cases}$$

$$z_t^{I_t} = f\left(z_{t-1}^{I_t}\right) + qw_t$$

$$X_t = M\sigma\left(z_t^{I_t}\right) + sv_t$$

Update on all attractor set

- Serial update
  - At each time, update state for only one intention
  - Update the state of the BAM corresponding to the intention with the largest probability
    - Intentions are updated according to the likelihood of active BAMs
  - Computation time and results vary depending on the order of active BAMs



$$I_t = \begin{cases} I_t & (with\ 1-p) \\ I \sim Mult(\boldsymbol{\alpha}). & (with\ p) \end{cases}$$

$$I_g = argmax_{I_t} P(I_t)$$

$$z_t^{I_g} = f\left(z_{t-1}^{I_g}\right) + qw_t$$

$$X_t = M\sigma\left(z_t^{I_g}\right) + sv_t$$

Update on one attractor set

Simulation of HBAM

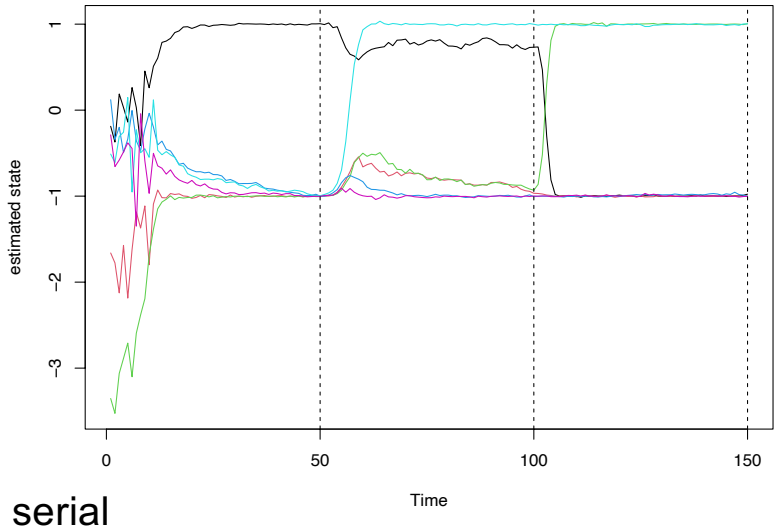
- Setting
  - In this evaluation, the behavior of HBAM was evaluated using artificially generated attractors
  - The representative value of each attractor is $\mu_i = (i, i+1, i+2)$
  - Attractor sets were created by clustering together representative values that are closer
  - The HBAM parameters were set to p=0.2, q=1, s=3
  - Observed values were given over 150 time slots, with the correct attractor set switching every 50 time slots
- Metric
  - **Correctness** is the percentage of time slots in which the decision state and the correct attractor coincide out of the 150 time slots
- Other method
  - **original BAM** is assumed to have the same attractors as HBAM but with a flattened hierarchy in a single attractor set.

Example of decision state

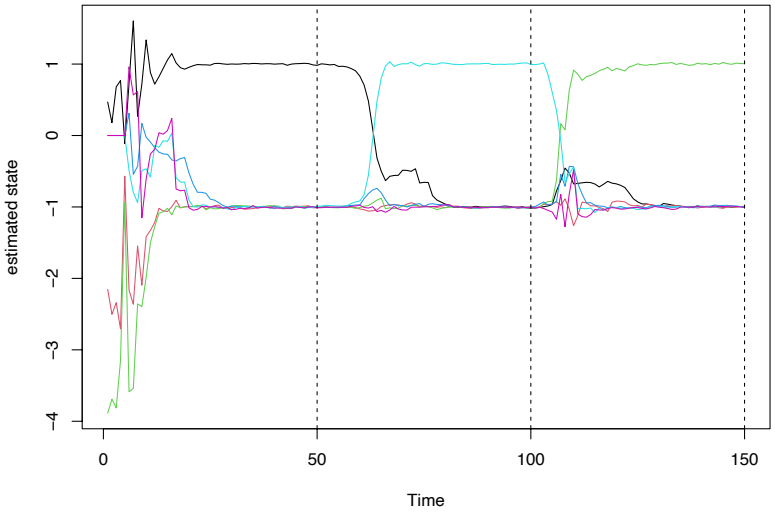
- Setting
  - 3 attractors x 2 sets
- Result
  - Both parallel and serial updates are able to change the decision-making state according to the changing observation values
  - The parallel update tends to be slightly faster because all attractor sets are updated with some degree of activity, which allows for faster response to changes that require attractor set switching
  - In the case of parallel updates, the attractor set is switched at the 100 time-slot and the previously high attractor is suppressed in favor of the new attractor.

parallel

Collect: black -> lightblue -> green

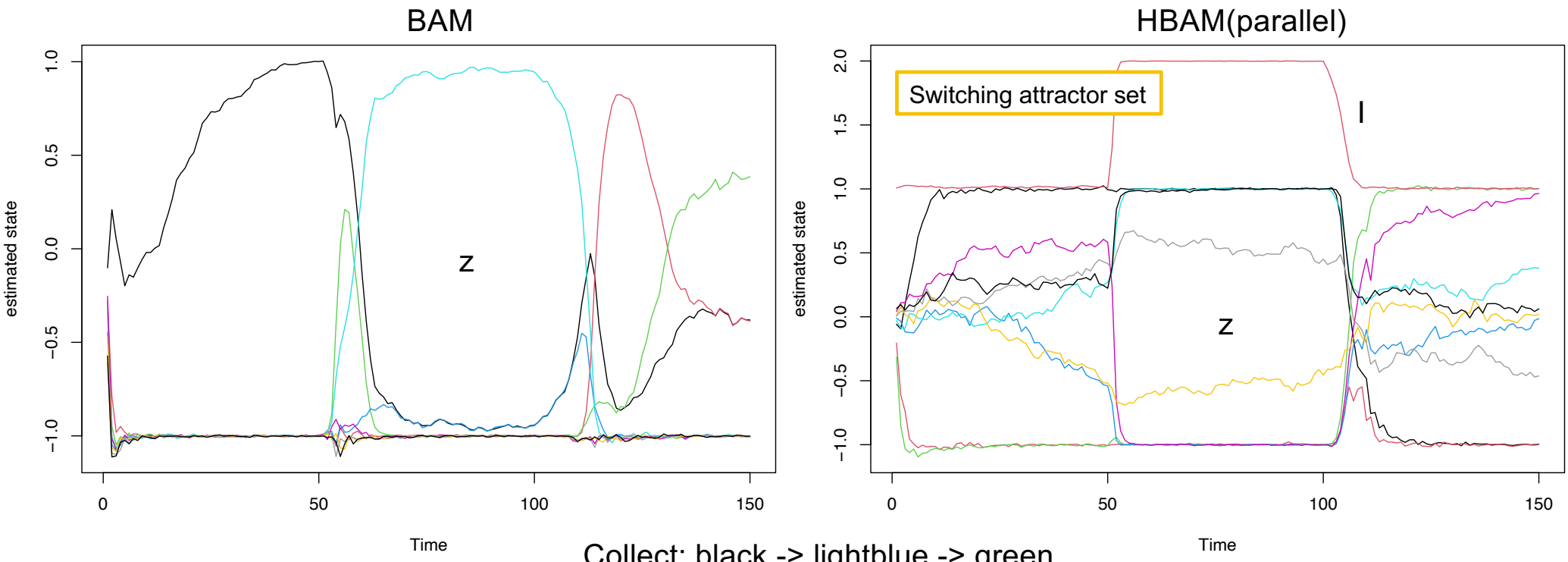


serial



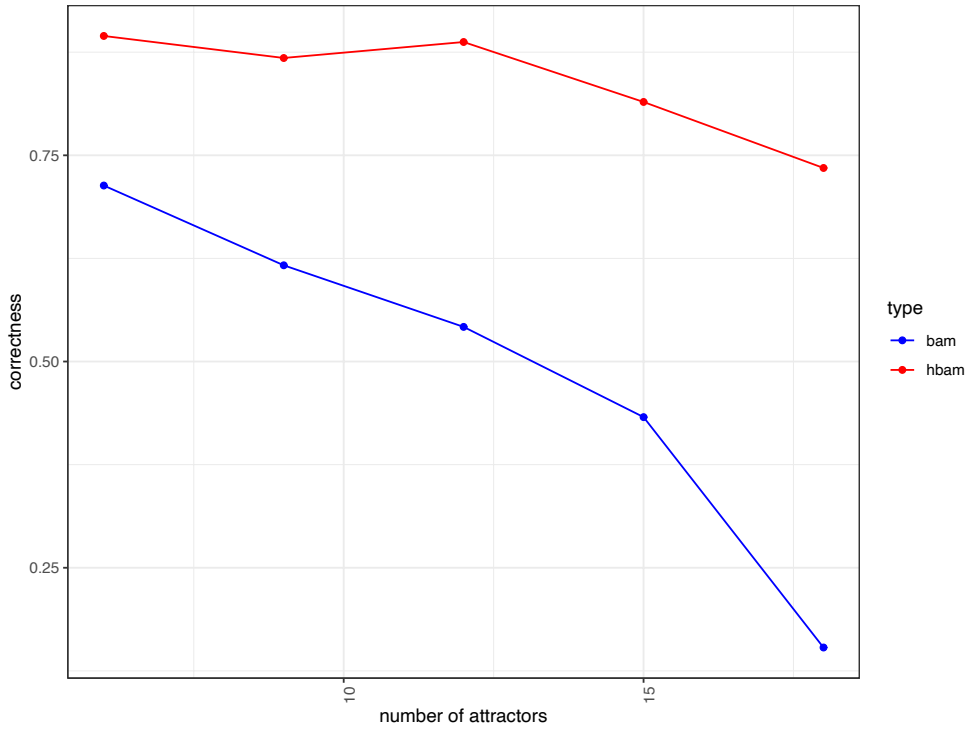
BAM v.s. HBAM

- BAM needed parameter tuning due to a change in the number of attractors
- Hierarchical BAM reached the correct attractor faster



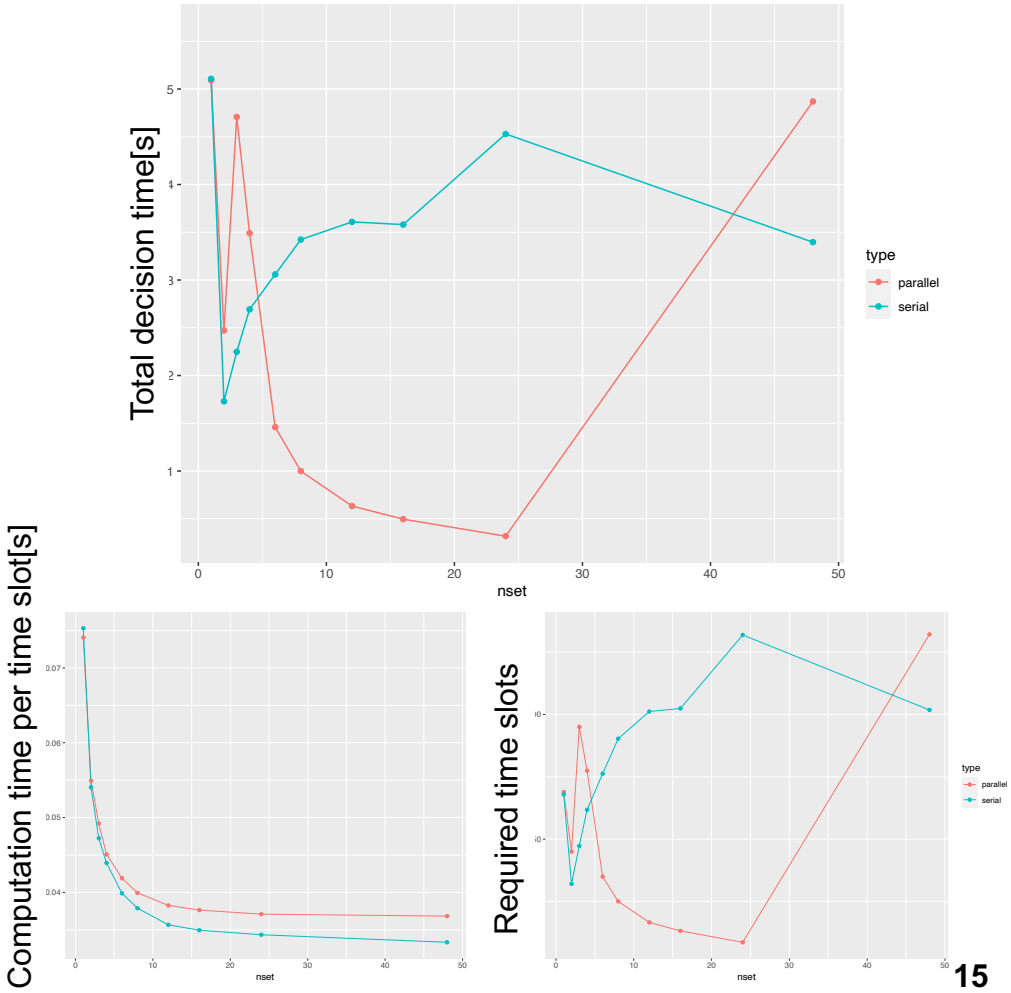
Correctness with number of attractors

- Setting
  - The percentage of correct responses was evaluated for the original BAM and HBAM as the number of attractors was increased to examine the effect of hierarchy
  - The number of attractors in the attractor set was fixed at 3 and the overall number of attractors was increased by increasing the number of attractor sets, with HBAM performing state updates using parallel update
  - The red line represents the results of HBAM and the blue line represents the results of BAM
- Results
  - The original BAM shows a sharp decline in the correctness rate as the number of attractors increases, while HBAM shows a slower decline in growth rate as the number of attractors increases
  - BAM is optimal as the number of attractors increases, but HBAM is able to increase the overall number of attractors while keeping the number of attractors in each set constant, making it a more scalable model for increasing the number of attractors.



Time required to decision with serial and parallel updates

- Setting
  - The impact of the number of attractor sets on decision-making was evaluated by changing the size of the split for a total of 48 attractors
  - The number of attractor sets was varied from 1 to 48
  - The red line indicates the parallel update case and the blue line indicates the serial update case
- Results
  - The computation time per time slot decreases as the number of attractor sets increases because the number of attractors per set decreases, which leads to reduced computation time
  - Overall, the computation time per time slot for serial update is shorter than for parallel update because only one set of attractors is active and computed in series update, but the difference is small
  - If the number of attractor sets with the shortest decision time is used, parallel update is faster than serial update in terms of both decision speed and computation time per time slot
- Based on these results, it is considered that hierarchical updating with parallel updating is suitable.



Summary

- In this paper, a hierarchical version of BAM (HBAM) was proposed to quickly determine solutions to task assignment problems in edge-cloud computing environments
- HBAM introduces a hierarchical structure into BAM and selects similar attractors from a small number of attractors in a set while switching between multiple attractor sets, allowing the system to maintain a high accuracy rate even as the total number of attractors increases
- The state update method and attractor set size associated with the hierarchical structure were evaluated and it was found that parallel updates and an intermediate size for the attractor set are preferable
- Future research will evaluate the performance of HBAM in a simulation environment that mimics an actual edge-cloud computing environment and examine the best clustering method for constructing the attractor set.