Optimizing IoT to Fog Latency in Resource Allocation

Ismael Martinez

March 2019

1 Problem Description

Concerning the vast amount of data that is being output by Internet of Things (IoT) devices, there exists the well-known Cloud and lesser known but emerging Fog Computing Paradigms [1] to facilitate storage and processing. In this project, I intend to define a model and seek a solution to the optimization of data distribution to Fog and Cloud devices in order to minimize latency of data processing and turnaround, constrained by the storage capacity of a fog devices for a given time. I have modeled the problem as a network with stochastic capacity at each fog device, and elect to use Dynamic Programming to solve the problem.

2 Mathematical Description

We define k to be the time steps at which we make our distribution decision regarding the network, $k = 0, \dots, N-1$ for N the total number of time steps. Referring to Figure 1, we see a set of nodes d_i representing IoT devices connected to a set of nodes f_j representing fog devices by a *latency cost* $l_{i,j}$ which represents the wireless travel time from node d_i to f_j and vice versa. A *package* $p_{i,k}$ is the data generated by d_i at time k with file size $m_i > 0$ that must be distributed to some node f_j.

We denote the total memory storage space of f_j as M_j. This can be seen in Figure 1 as the capacity of each node f_j. The cloud node $C = f_0$ has a memory value of ∞ since for our purposes we say it has no memory limit. In addition to a storage limit, each fog node f_j has a congestion factor and variance $\Delta_j^0, \sigma_j^0 \in [0, 1]$ that describe the memory needed by data coming in from outside this system, and a processing factor and variance $\Delta_j^1, \sigma_j^1 \in [0, 1]$ that describe the amount of data released by node f_j. The amount of extra data entering f_j follows a normal distribution $\Delta_{k,j}^0 \sim \mathcal{N}(\Delta_j^0 \cdot M_j, \sigma_j^0 \cdot M_j)$, $j \neq 0$; the amount of data released from storage of f_j follows a normal distribution

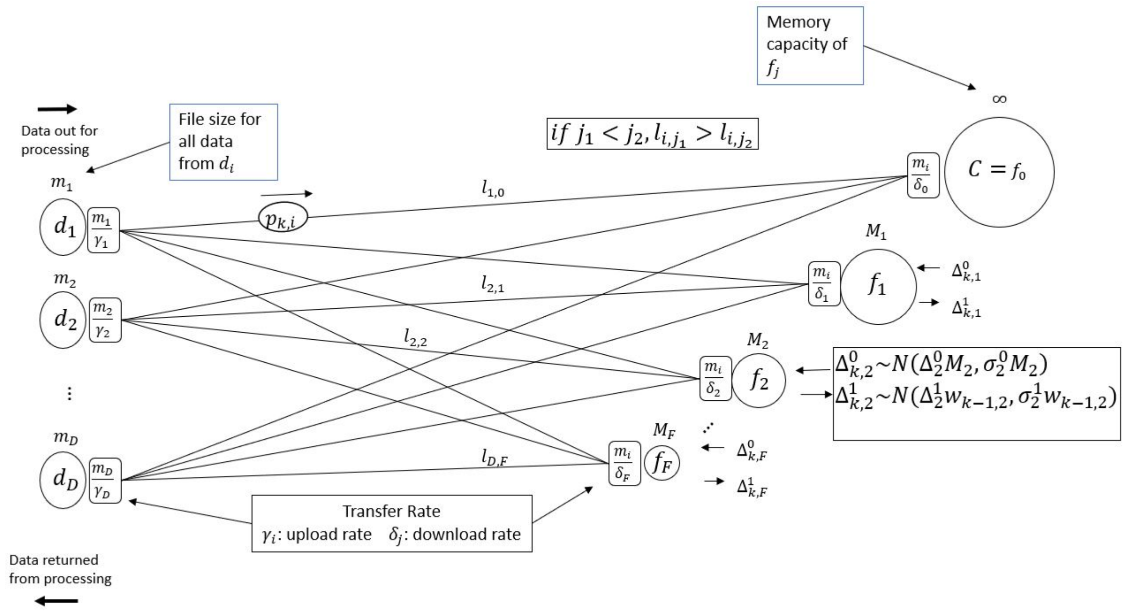

Figure 1: The network architecture of the IoT to Fog Latency Problem. Physically, graphically and mathematically, the distance between the device and each f_j decreases as j increases. At each k and for each i, we send a package $p_{k,i}$ from each d_i to $u_{k,i} = f_j$. Each package $p_{k,i}$ has a file size of m_i, transfers out of d_i with time $\frac{m_i}{\gamma_i}$ and down to f_j with time $\frac{m_i}{\delta_j}$. Each fog device f_j has a max storage capacity of M_j, a congestion factor of Δ^0_j, congestion variance σ^0_j, processing factor of Δ^0_j and processing variance σ^1_j.

.

$\Delta^1_{k,j} \sim \mathcal{N}(\Delta^1_j \cdot w_{k-1,j}, \sigma^1_j \cdot w_{k-1,j})$, $j \neq 0$. We then have

$$w_{k,j} = \max\{0, w_{k-1,j} + \Delta^1_j - \Delta^0_j - \sum_{i \in I} m_i\}$$

units of storage available at f_j at time k; this is the previous storage amount with an amount of processed data released that free up storage, and an amount of data storage congestion subtracted. The maximum storage, congestion and processing factors are shown in Figure 1. One implication of the transition, given that $w_{k,j}$ cannot exceed M_j, is that

$$\sum_{i \in I} m_i \leq \min(w_{k-1,j}, M_j\}$$

for the set I of devices d_i who send a package to f_j.

We define the upload rate of d_i to be γ_i, and the download rate of f_j to be δ_j. Since the file size of $p_{k,i}$ is m_i, then the *upload time* of d_i is $\frac{m_i}{\gamma_i}$, and the *download time* of f_j is $\frac{m_i}{\delta_j}$. Since the data sent by the IoT devices is relatively small in size and can be processed rapidly on most common computers, for our purpose we say the processing time is negligible. Similarly, we assume the response message size is small (under 1 MB) and can be ignored. In other words, for the response from f_j to d_i, we only care about the latency cost $l_{i,j}$. In total, to time to send a package $p_{k,i}$ from d_i to f_j and back is

$$\frac{m_i}{\gamma_i} + 2l_{i,j} + \frac{m_i}{\delta_j}$$

2.1 Assumptions

We make three assumptions concerning the location of devices and time steps k of our problem.

- We assume that all IoT nodes d_i exist in some cluster of radius r, where $r < l_{i,F}, \forall i$. Recall that f_F is the closest fog device to our set of IoT device nodes. This is shown in Figure 2.
- $\forall j$, l_j is smaller than the time in a single time step $k-1$ to k.
- If $j_1 < j_1$, then $l_{i,j_1} > l_{i,j_2}$. This means we order the f_j is order of decreasing distance.

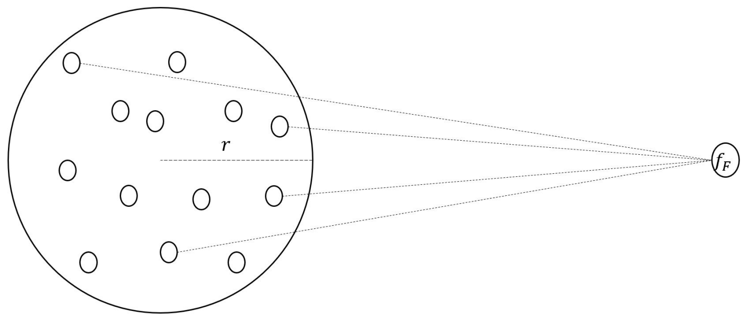

Figure 2: All IoT devices reside in some cluster of radius r, where $r < l_{i,F}, \forall i$
.

2.2 Model Description

Our objective is to minimize the total lifecycle of each package $p_{i,k}$ and its response from d_i to some node f_j and back, while incurring latency cost $2l_{i,j}$ plus transfer costs of $\frac{m_i}{\gamma_{k,i}} + \frac{m_i}{\delta_{k,j}}$.

- The *state* x_k is the description of the available storage at each node f_j at time k, where $x_k = w_{k-1} = \{w_{k-1,1}, w_{k-1,2}, \dots, w_{k-1,F}\}$.
- *decision* $u_{k,i}$ is the choice of fog device f_j to which to send $p_{k.i}$; the kth system decision is $u_k = \{u_{k,1}, u_{k,2}, \dots, u_{k,D}\}$.
- The *cost* of each $u_{k,i}$ is

$$g_{k,i}(, x_k, u_{k,i}) = \frac{m_i}{\gamma_i} + 2l_{i,j} + \frac{m_i}{\delta_j}$$
$$g_k(x_k, u_k) = \sum_{i=1}^{D} g_{k,i}(x_k, u_{k,i})$$

- The *transition function* is

$$f_{k,j}(x_k, u_k) = \max\left\{0, w_{k-1,j} + \Delta^1_{k.j} - \Delta^0_{k,j} - \sum_{p \in P_{k.j}} m(p)\right\}$$
$$f_k(x_k, u_k) = \{f_{k,1}, f_{k,2}, \dots, f_{k,F}\}$$

An exact solution to find an optimal policy uses a backward chain method beginning at $g_N(x_k) = J_N(x_k) = 0$. At each step k and state x_k, we evaluate the cost

$$J_k(x_k) = \min_{u_k \in U(x_k)} \{g_k(x_k, u_k) + J_{k+1}(f_k(x_k, x_k, w_k))\}$$
$$\mu_k(x_k) = \arg\min_{u_k \in U(x_k)} \{g_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k, w_k))\}$$

where $J_k(x_k)$ is the cost from k to N of making an optimal decision at state x_k. We note that the cost does not depend on w_k for our system.

For an approximate solution, we use a base approximation $\tilde{J}_k(x_k)$ to denote an approximated cost from k to N. In this case we use a forward chain with a one-step lookahead to approximate the cost

$$\hat{J}_k(x_k) = \min_{u_k \in U(x_k)} \{g_k(x_k, u_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}$$
$$\hat{\mu}_k(x_k) = \arg\min_{u_k \in U(x_k)} \{g_k(x_k, u_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}$$

where $\tilde{J}_k(x_k)$ is a parametric approximation of $J_k(x_k)$. We go into further detail of the parametric approximation in §2.3.

2.3 Parametric Approximation

The chosen approximation was that of Parametric approximation where we approximate $J_k(x_k) \approx \tilde{J}_k(x_k) = \sum_{i=0}^{2S+3} y_i(x_k) r_i$. The features chosen are $y_i(x_k) = k^i$ for $i = \{0, 1, 2\}$, $y_{3+i}(x_k) = s_i$ and $y_{S+3+i}(x_k) = s_i \cdot k$ for each state; $s_i \in \{0, 1\}$ represents whether x_k is in state s_i or not, with $|X_k| = S$. The Exact Solution algorithm is first run for a small number of steps. This gives us a set of costs for each state and step k. To obtain optimal values r_i, I fit the features to a linear regression against the cost obtained from the Exact Algorithm, which is equivalent to minimizing the squared distance $\sum_{i=0}^{2S+3} |y_i(x_k) r_i - t_{k,i}|^2$ for the target cost $t_{k,i}$ at step k. With this base approximation

3 Results

3.1 Simulated Data

A data set was simulated with 2 IoT devices and 3 fog devices (including 1 Cloud) using the call `python FogSimulateData.py -d 2 -f 3 -s 13` – that's 2 IoT devices, 3 fogs and a seed of 13. The device data included a memory size for each IoT device, an upload rate for each device, as well as a location (x, y) pair. The fog data included a storage limit, a congestion mean and variance, a processing mean and variance, a download rate, a distance from the origin, and a location (x, y) pair. The latency between IoT devices and fog devices was calculated from these location pairs. Although these parameters seem small, this problem gets quite large quite quickly. For example, the number of decisions is F^D for D IoT devices and F fog devices, and the number of states can be at most $2^{D^{F-1}}$ (§ 3.3).

3.2 Instances

To recreate this project instance, I used a seed of 40 for both. Using the simulated data produced by `FogSimulateData.py`, I ran the following two scripts:

- `python FogExactStochasticMemory.py -d <path_to_simulated_device_file> -f <path_to_simulated_fog_file> -s 40 -N 4`

- `python FogApproximationStochasticMemory.py -d <path_to_simulated_device_file> -f <path_to_simulated_fog_file> -s 40 -N 10 -o <path_to_optimal_solution_from_exact>`

In the above, the `-o` parameter is the solution file produced by the Exact Algorithm. Both instances had $|X_k| = 16$ states per stage k, and $|U_k(x_k)| = 9$ possible decisions per state x_k. The Exact Algorithm took 8 seconds for $k = 0, ..., 4$ while the Approximation 45 seconds for $k = 0, ..., 10$ steps. Each step of the Exact Algorithm was about 2 seconds, whereas each step of the Approximate Algorithm was about 5 seconds; this is due to the longer nature of predicting on the fitted model, which has $2S + 3 = 35$ features. That being said, since we are doing a 1SL solution, the approximate algorithm is still more beneficial to compute a single estimated step at a given $k > 2$ for $N > 5$ The absolute average percent error of our fitted model to approximate $\tilde{J}_k(x_k)$ was $\frac{1}{T}\sum_{j=1}^{T}|\frac{\sum_{i=0}^{@S+3} y_i(x_k)r_i - t_j}{t_j}| = 0.01723$ and an R-squared of 1.0.

The two models were compared with the call `python FogExactApproximativeComaprison.py -e <path to optimal_solution.csv> -a <path to optimal_solution_approximate.csv>`. There is no seed needed for this comparison as it just takes in the optimal solution files from each of the runs.

3.3 Memory Thresholds

In order to minimize the number of states, I used memory thresholds to define the states and used a Normal Distribution CDF to find the probability of a fog attaining that threshold given the previous threshold, the current decision, and the congestion and release of data. For example, if I had two fog devices with file sizes of $m_1 = 2$ and $m_2 = 3$, then I will only be interested on whether a fog can take either or both of our packages. The possible memory thresholds for a fog device would then be {0, 2, 3, 2+3=5}. For D devices each with file size m_i, we will have a possible threshold size for a single fog device of $\sum_{n=0}^{D}\binom{D}{n} = 2^D$. Then, for $F - 1$ non-cloud devices, we will have $2^{D^{F-1}}$ possible states.

3.4 Optimal Policy

The policy decision and cost between the Exact Solution and the 1SL are quite similar. With $D = 2$ and $F = 3$, our simulated data gives us 16 states;

State - Memory currently available at fogs	Exact - Optimal Policy	Approximate - Optimal Policy
(inf, 0, 0)	(0, 0)	(0, 0)
(inf, 0, 1)	(0, 2)	(0, 2)
(inf, 0, 2)	(2, 0)	(2, 0)
(inf, 0, 3)	(2, 2)	(2, 2)
(inf, 1, 0)	(0, 1)	(0, 1)
(inf, 1, 1)	(0, 2)	(0, 2)
(inf, 1, 2)	(2, 1)	(2, 1)
(inf, 1, 3)	(2, 2)	(2, 2)
(inf, 2, 0)	(1, 0)	(1, 0)
(inf, 2, 1)	(1, 2)	(1, 2)
(inf, 2, 2)	(2, 1)	(2, 1)
(inf, 2, 3)	(2, 2)	(2, 2)
(inf, 3, 0)	(1, 1)	(1, 1)
(inf, 3, 1)	(1, 2)	(1, 2)
(inf, 3, 2)	(2, 1)	(2, 1)
(inf, 3, 3)	(2, 2)	(2, 2)

Table 1: The optimal policy μ_1^* for different amounts of memory available at time $k = 1$. A decision (x_1, x_2) is the choice j for each of device d_1 and d_2. e.g. (2, 1) means d_1 sends its package to f_2, and d_2 to f_1.

the optimal policies from the exact solution and the approximate solution are shown for $k = 0$ in Table 1. The full set of solution is in the attached files `optimal_solution.csv` for the exact solution and `optimal_solution_approximate_regression.csv` for the approximate solution. This problem has the same number of states at each step, with the same probabilities of moving to different states for a given action; therefore, the optimal policy is only dependant on the state (storage available) and not the step.

For the costs, we do one test to compare the costs followed by one test to predict future costs. In Figure 3, we see how closely the the exact cost $J_k(x_k)$ is to the approximated cost $\hat{J}_k(x_k)$, for $x_k = (\text{inf}, 1, 1)$, the units of storage available at each fog device. The full solutions for both the exact and the approximative solutions are available in `optimal_solution.csv` and `optimal_solution_approximate_regression.csv`.

The second test we perform is using this well fit model to extrapolate to more steps. We have fit the model on $N = 4$ steps, and we extend the model to $N = 10$ steps. Since we are estimating the cost from k to the end, we need to shift the new model so that $J_k(x_k) \approx \hat{J}_{k+10-4}$. We see in Figure 4, this shift is shown with the last 5 points of the approximate cost. We then extend the model backwards to $k = 0$. This type of analysis shows the usefulness that a 1SL approximation could have – not only with attaining an immediate estimate $\hat{J}_k$, but also with modifying the parametric approximation to larger values of N. The full solution set is in the attached file `optimal_solution`$_{a}pproximate\_extension.csv$.

References

[1] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, Longxiang Gao, Yong Xiang, and Rajiv Ranjan. Fog computing: survey of trends, architectures, requirements, and research directions. *IEEE access*, 6:47980--48009, 2018.

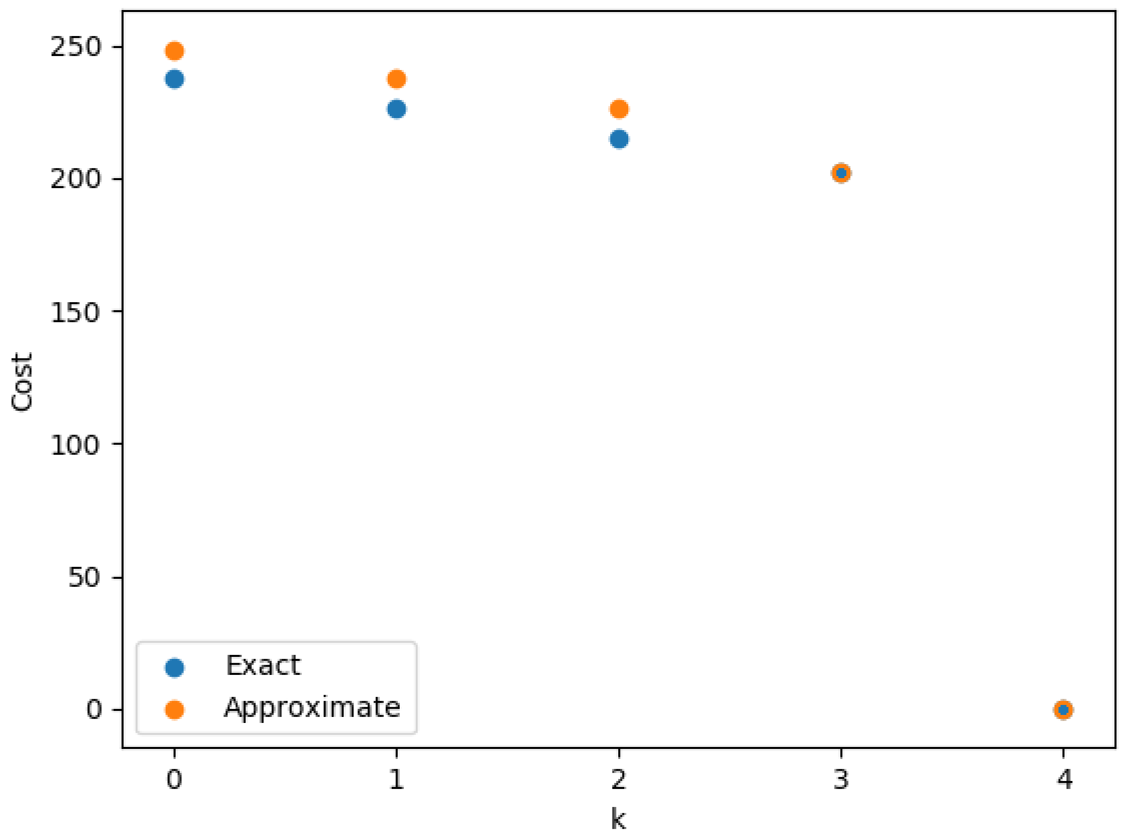

Figure 3: Comparing the Exact Solution costs with the Approximated 1SL solution costs for $N = 4$. The cost of the final step J_N is 0 for both exact and approximative costs. These are the costs associated with the state (inf, 1, 1).

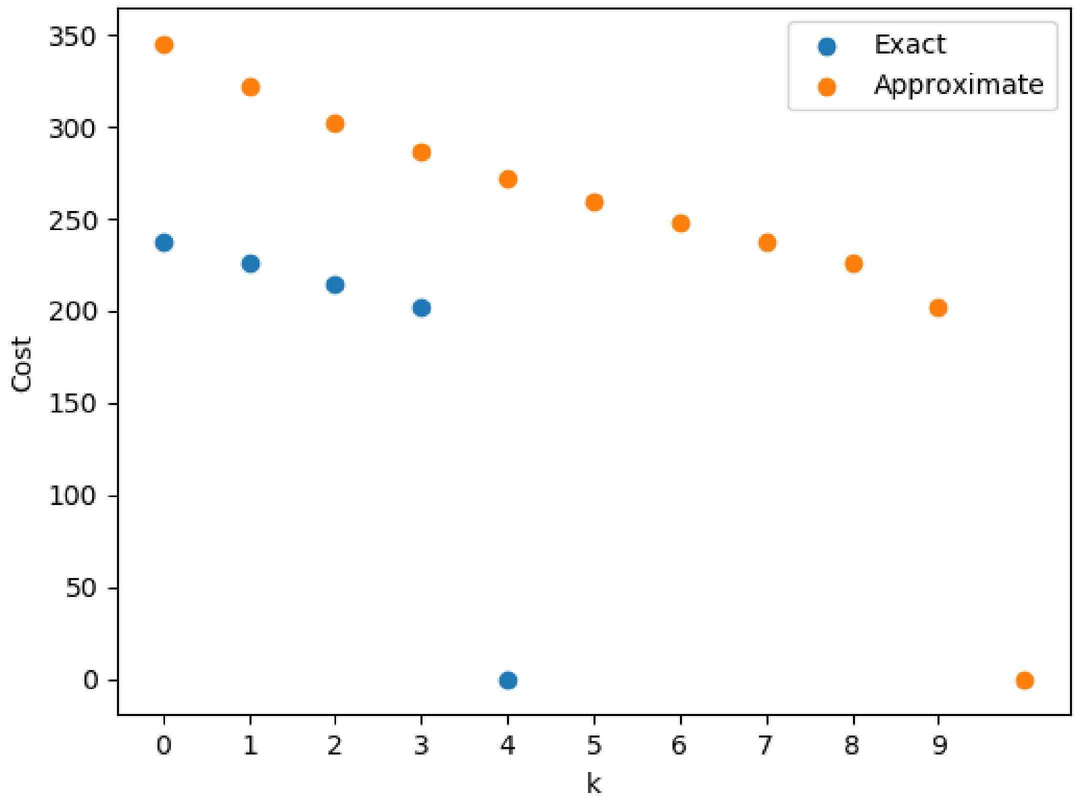

Figure 4: We extend the trend of the fitted line from the Exact Solution data of $N = 4$ to a approximated problem with $N = 10$. These are the costs associated with the state (inf, 1, 1).