**Requirements**

1. Need a system with at least python3.6
2. In order to run docker experiments we need to have docker installed with user permissions.
   1. <https://docs.docker.com/engine/install/ubuntu/>
   2. <https://docs.docker.com/engine/reference/commandline/docker/> (Useful Docker commands)
3. Python packages and dependencies present in requirements.txt
   1. pip install -r requirements.txt

**Setup & Configuration (Docker testbed)**

1. Go to inop\_cloud folder
2. Make sure to execute the commands below before running other docker images.
3. Execute the following commands:
   1. **docker build --tag inop .**
   2. **docker run -d --name=app1 -p 4000:3333 inop**
4. Go to docker\_dummy subfolder
5. Execute the following commands:
   1. **docker build --tag inop\_dummy .**
   2. **docker run -d --name=app2 -p 5000:3333 inop\_dummy**
6. Come out of the docker\_dummy folder to main folder (inop\_cloud)
7. Execute the following commands to check if the IP address of the edge nodes match the default IP address:
   1. **docker inspect -f '{{ .NetworkSettings.IPAddress }}' app1** (should be 172.17.0.2)
   2. **docker inspect -f '{{ .NetworkSettings.IPAddress }}' app2** (should be 172.17.0.3)

**Running a Scenario**

A. **Computer Simulations**

**Training**

(SALMUT)

Scenario 1

*python main\_train.py --algo 3 --folder scenario\_1\_lambd\_2 --env\_name salmut --logdir salmut\_log --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve False*

Scenario 2

*python main\_train.py --algo 3 --folder scenario\_2\_lambd\_2 --env\_name salmut --logdir salmut\_log\_2 --lambd 0.5 --lambd\_evolve True --user\_identical True --user\_evolve False*

Scenario 3

*python main\_train.py --algo 3 --folder scenario\_3\_lambd\_2 --env\_name salmut --logdir salmut\_log\_3 --lambd 0.5 --lambd\_evolve True --user\_identical False --user\_evolve False*

Scenario 4

*python main\_train.py --algo 3 --folder scenario\_4\_lambd\_2 --env\_name salmut --logdir salmut\_log\_4 --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve True*

Scenario 5

*python main\_train.py --algo 3 --folder scenario\_5\_lambd\_2 --env\_name salmut --logdir salmut\_log\_5 --lambd 0.5 --lambd\_evolve True --user\_identical True --user\_evolve True*

Scenario 6

*python main\_train.py --algo 3 --folder scenario\_6\_lambd\_2 --env\_name salmut --logdir salmut\_log\_6 --lambd 0.5 --lambd\_evolve True --user\_identical False --user\_evolve True*

For PPO and A2C, replace –algo with 0 and 1 respectively, and –env\_name with ppo and a2c and –logdir names accordingly.

**Evaluation** (This can be run in parallel or serially. Can only be performed once training is done.)

Let’s look at an example of parallel evaluation of eval for SALMUT.

*python main\_eval2.py --algo 3 --folder scenario\_1\_lambd\_2 --env\_name salmut --logdir s1 --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve False --start\_iter 0 --step 333*

*python main\_eval2.py --algo 3 --folder scenario\_1\_lambd\_2 --env\_name salmut --logdir s1 --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve False --start\_iter 333 --step 333*

*python main\_eval2.py --algo 3 --folder scenario\_1\_lambd\_2 --env\_name salmut --logdir s1 --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve False --start\_iter 666 --step 334*

Similarly, we run it for different scenarios by changing the values of (user\_identical, user\_evolve, and lambda\_evolve).

We can also run similar experiments for PPO and A2C by changing algo, env\_name, and logdir similar to training.

**Planning**

We run the planning algorithms (DP and baseline) using the following commands:

Dynamic Programming Solution (algo = 0)

*python main\_plan.py --algo 0 --folder scenario\_1\_lambd\_2 --env\_name plan\_eval --logdir plan\_log\_s1 --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve False*

Baseline Policy (algo = 1)

*python main\_plan.py --algo 1 --folder scenario\_1\_lambd\_2 --env\_name thres\_eval --logdir thres\_log\_s1 --lambd 0.5 --lambd\_evolve False --user\_identical True --user\_evolve False*

**B. Behavioral Analysis (Computer Simulations)**

Only need to run training, commands are the exact same as the training for computer simulations. Only thing is we need to execute them in inop\_salmut\_behavior folder.

**C. Docker testbed**

Each experiment needs to be run in separate VM. Execute the command in screen. We run only for SALMUT and Baseline. The buffers folder for each scenario should have the files N.npy and lambda.npy.

scenario\_1\_lambd\_2/buffers/N.npy

scenario\_1\_lambd\_2/buffers/lambda.npy

**Training**:

*python3.6 load\_gen.py scenario\_1\_lambd\_2 salmut*

*python3.6 load\_gen.py scenario\_2\_lambd\_2 salmut*

and so on …

**Evaluation**

Change app2.py with app2\_eval.py in Dockerfile and perform the steps in Configuration & Setup. Ensure all training data is present in buffers, if we had run different seeds of training in separate VMs copy all the data to all the machines, we are going to run evaluation in.

Make sure of the following:

1. Stop and delete any existing docker images that may be running by using:
   1. docker stop <container-name>
   2. docker rm <container-name>
2. The results folder in the scenario\_folder is empty unless the goal is to restart training from previously stopped iteration.
3. If we restart training, make sure to restart the edge node using:
   1. docker restart app1

*python3.6 load\_gen\_eval.py scenario\_1\_lambd\_2 salmut\_eval*

**Baseline**

Change app2.py with app2\_baseline.py in Dockerfile and perform the steps in Configuration & Setup.

*python3.6 load\_gen\_.py baseline.py scenario\_1\_lambd\_2 salmut\_eval*

If you are running the experiments using python command directly instead of python3.6, replace python3.6 by python or python3 in load\_gen.py, load\_gen\_eval.py and load\_gen\_baseline.py in the *run\_rl\_module\_and\_notify()* function.

**Generating Results**

**A. Computer Simulations**

If we run evaluations in parallel, combine the results of the experiments before we plot the results:

*python combine\_result.py <scenario-folder> <algorithm>*

Ex:

*python combine\_result.py scenario\_1\_lambd\_2 salmut*

Plotting results (ensure file names are same as in results folder of the scenario)

*python plot\_results.py scenario\_1\_lambd\_2*

The results are in the scenario folder under results directory with name {folder}\_reward.png, ex: *scenario\_1\_lambd\_2\_results.png*

**B. Computer Simulations (Behavioral Analysis)**

Plot the results

*python plot\_results.py scenario\_1\_lambd\_2*

The results are in the scenario folder under results directory with name {folder}\_offload\_new.png and {folder}\_overload\_new.png.

*python plot\_pareto.py scenario\_1\_lambd\_2*

The results are in the scenario folder under results directory with names pareto\_{folder}.png

**C. Docker Testbed**

If we run only training and want to see the results. All the buffers files of the different training runs needs to be in the same scenario folder.

Combine the results using

*python combine\_results.py scenario\_1\_lambd\_2*

Plot results using (plots offload, overload counts as well)

*python plot\_results\_combined.py scenario\_1\_lambd\_2*

The results are in the scenario folder under results directory with names {folder}\_dis\_reward\_testbed.png, {folder}\_offload\_testbed.png, and {folder}\_overload\_testbed.png.

*python plot\_pareto.py scenario\_1\_lambd\_2*

The results are in the scenario folder under results directory with names pareto\_testbed\_{folder}.png