Reinforcement Learning and Dynamic Optimization

Project 1 - Phase 1

George Konofaos: 2018030175

Task 1:

State Space: In order to be able to describe the state space we need to know in

1) which physical node each VNF is executed
2) the markov state of Demands (H , L)

So the Possible states are described by: $St = \{node_i(t), demand_i(t)\}_{i=1,2,...,N}$

For the simple example of 2 Physical nodes and 2 VNFs there is a total of 16 states.I have placed a legend at the end of this report explaining both The actions and the states in my simulation.

Action Space: At time t we assign the VNFs on any physical node M so the Action space is described as $At = \{node_i(t + 1)\}_i=1,2,...,N$, since we can assign Each N to each M we have M^N possible actions.

Transition Probabilities:

Given the current state and action, the transition to the next state depends on:

$P(si'=H \mid si=H)=PHHi$,

$P(si'=L \mid si=H)=PHLi$

$P(si'=L \mid si=L)=PLLi$,

$P(si'=H \mid si=L)=PLHi$

The new placement of VNFs is determined by the action a.

In this environment the markov chain only cares about the demands and the locations do not affect it , so for any given location these are the transition probabilities.

Rewards:

The rewards can be calculated as the negative of the cost, if we try to maximize the reward , we will minimize the cost.

- Power Cost: C_{on} = c_{on} X # of active nodes
- Reconfiguration Cost: C_R = c_R X # of VNFs moved
- SLA Violation Cost: $C_{SLA} = \Sigma^{M}_{J=1}\Sigma_{i:li=j}\max(0,\Lambda^{j}(t)-c) \text{ X } c_{SLA} \text{ X } (\Lambda^{j}(t) - C)^{2}$

Total reward: R(s,a)=−(w_ON×C_{on} +w_R×C_R +w_SLA×C_{SLA})

Terminal States:

There are no terminal states in this problem.

Questions to answer:

1) Assume N = M = 2, and the capacity of each node C = 1.2. Find a set of parameter values w_{ON}, w_{R}, w_{SLA} for which the optimal policy is to always assign all VNFs to a single physical node (let's call this Group-ALL policy).

   We need to configure the weight variables so that it is more costly to keep multiple physical nodes active.
   For example we could set the weights: w_ON =10 , w_R=1 , w_S=1

   And we expect from this experiment to see only actions 0 (0,0) or 3 (1,1)

   Result: Optimal Policy: [0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 3] for every possible state we get an expected action of either 0 or 3 as expected.

2) N = M = 2, C = 1.2. Find a set of parameter valueswON , wR, wSLA for which the optimal policy always keeps all M nodes ON (Split-ALL policy):

   In this case we need to configure the weight variables so that it is preferable to keep VNFS in different physical nods.

   For example we could set the weights: w_ON =1 , w_R=1 , w_S=10

   We expect that in each state the best action is either 1 or 2 but there is a problem for the states marked with demands (L L) and they are on the **same** node, my algorithm struggled with these states but the closest i made it to split-all policy was this result.

Optimal Policy: [1 1 1 0 1 1 1 1 2 2 2 2 1 1 1 3]
(every 4th state indicates a state with demands LL)
There wasnt a weight that i could influence so that in states where we are in the same node with demands LL so that it choose to change them.
Either i increased the w_ON in which case it prefers to keep them in the same so it wouldnt change the above result.
Either i increased the w_R in which case it was more costly to change it so it kept it the same.

3) For the above parameters (N=M=2, C= 1.2), demonstrate that Policy Iteration indeed solves the problem optimally (try both “corner-case”' or “toy”' scenarios like the above, as well as more generic ones).

   First Senario i tried was also a corner-case with weights
   w_ON =1 , w_R=10 , w_S=1.

   We expect that with this configuration we will see as a result that we dont change the VNFs from the node they already are.

   Optimal Policy: [0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3]

   This is the result we got and it confirms our estimates, states from 0 to 3 are the states where both VNFs are in node 0.

   Second Senario this time i decided to put similar weights and see how the algorithm would define the optimal policy.Note: this configuration took the most to run from all the other cornercases.

   w_ON =1 , w_R=1, w_S=1.

Optimal Policy: [0 0 0 0 1 3 3 3 3 3 3 3 3 3 3 3]

From this we can derive that the algorithm prefers having almost all of the Time prefers to keep both VNFs in one node to minimize the C_on cost. Also it seems that it prefers to take the SLA cost in states 3 and 11 than to Change the VNFs to separate nodes since that would require to Reconfiguration cost and ON cost which would be bigger than the sla cost.

I also tried this scenario to put more emphasis that we didnt want SLA Violation.
w_ON =1 , w_R=1, w_S=5

Optimal Policy: [2 0 0 0 1 3 0 3 2 0 0 2 2 3 3 3]
^ ^

Here we can see that in in the states marked up which would cause SLA violation we see that it actually changes VNFs to seperate nodes, Also in states 1 to 3 it decides to keep them in then same node to avoid Reconfiguring them and the extra cost of running on 2 nodes, the same That happens to nodes 13 -15.

4) Try to increase N,M or both (but make sure N >= M). What is the larger scenario you can solve in less than 1h in Colab?

Colab seemed to be running slower than my laptop so i decided to Locally. Sadly my algorithm doesnt seem to be time efficient since even running N=3 and M=2 was alot slower and the maximum i could run under and hour was N=3 and M=3, and certainly the complexity in the states Grows exponentially.

Here i will type for the experiment N=2 and M=2 the states and their indexes:

((0, 0), ('H', 'H')): 0
((0, 0), ('H', 'L')): 1
((0, 0), ('L', 'H')): 2
((0, 0), ('L', 'L')): 3
((0, 1), ('H', 'H')): 4
((0, 1), ('H', 'L')): 5
((0, 1), ('L', 'H')): 6
((0, 1), ('L', 'L')): 7
((1, 0), ('H', 'H')): 8
((1, 0), ('H', 'L')): 9
((1, 0), ('L', 'H')): 10
((1, 0), ('L', 'L')): 11
((1, 1), ('H', 'H')): 12
((1, 1), ('H', 'L')): 13
((1, 1), ('L', 'H')): 14
((1, 1), ('L', 'L')): 15

Actions and their index:
[(0, 0):0, (0, 1):1, (1, 0):2, (1, 1):3]