Contributions on Solving COVID-19 Crisis with Reinforcement Learning

Tonmoay Deb

May 2021

1 Problem Definition

COVID-19 [3] Pandemic has emerged as one of the most contagious disease in these years. It has cost so many lives and put harsh effect on the world economy. Since it had became deadlier from the first quarter of 2020, it is still affecting hard, e.g., India recently (2021) [1] with newer variants amid several vaccines have been rolled out. Since 2020, there have been several lockdown attempts taken by the Governments worldwide to slow the COVID spread. To simplify, if we consider that the Government can take only two action, either open or close, based on the possible reward (how the number of cases are changing), the actions are about to switch between the states. These lockdown cases are still congruent as the cases are rising. So, in this work, we plan to evaluate the valuation of being in each state based on the consequences from data. In Reinforecement Learning, we can formulate this problem as a Markov Decision Process (MDP), where we represent the overall scenario as Tuple: $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Here $\mathcal{S}$ represent the states that was induced by the Governments. $\mathcal{A}$ is binary as discussed earlier. Each action comes up with a reward r, which determined what action we may take later. However, the Transition Probability P is still a challenging problem as the true transition between the states re unknown. So, we also, plan to tacke this issue using *Robust Optimization*. γ is the scalar value for discounting rewards over time. The fundamental goal of all these works is to come up with a policy π to take prompt decision of taking actions from the observations as

$$\pi* = \arg max_{\pi} E\left[\sum_{t} R_t(S_t, A_t, S_{t+1})|\pi\right]$$

. Initially, we solve the problem of computing optimal policy by using *Value Iteration* as discussed below.

2 Value Iteration

Value Iteration indicated the value of being in one particular state s in the state space $\mathcal{S}$. The value is updated in each grid space, MDP for each state s' over time. The update continues recursively until it reaches to the end state. This recursive process is also known as *Bellman Update*, which indicates the sum of expected reward of being in current state over the horizon. To give higher weight to the close states and lower to the further, the discount function $\gamma in 0, 1$ is chosen from the context of the problem. For example, $\gamma = 0$ defines that the value function does not look into the future, whereas $\gamma = 1$ is too futuristic. The overall update operation is illustrated in Equation 1.

$$V(s) = \max_{a} \sum_{s'} P(s'|a, s) \left[R(s, a, s') + \gamma V(s')\right] \tag{1}$$

The update is expected to return the optimal value, which is used to calculate the optimal policy π^* at that state s as:

$$\pi^*(s) = \arg max_{a} \sum_{s',r} P(s', r|s, a)[r + \gamma V(s')] \tag{2}$$

We have successfully implemented this for our *COVID MDP* problem and achieved a tentative optimal policy. However, the policy we had was not satisfactory because the transition in the dataset was somewhat uncertain. To resolve the limitation, we later focused on the problem of computing the best possible transition probabilities from ambiguous (random) one by approximating a *S-rectangular* set and mapping them for our problem. We have implemented Soft-Robust Mixed Integer Linear Programming, namely SR-MILP [2] to achieve that. We discuss the process in the next section.

3 Robustness in Markov Decision Process

Regular infinite horizon MDPs include a finite state-action space with stable transition probabilities and rewards. Solving that MDPs is relatively simpler and could be easily solved by regular methods, e.g., Value Iteration. However, for several real-world scenarios, the reward and transition probabilities parameters come in uncertain states, where robust optimization is required. For example, in our *COVID MDP*, we expect that the probabilities and rewards might change based on other consequences, e.g., Government Rules. For that purpose, it is possible that there would be more than one MDPs with similar state-action pairs, but different transition probabilities and rewards. To find out a robust policy out of these uncertainties, we utilize Soft-Robust Optimization [2], which maps the uncertain parameters into a generalizable *S-Rectangular* set and optimized the worst-cases scenario from this set. More specifically, this technique relies on conditional value at risk based on level $\alpha \in [0, 1]$. The overall soft-robust objective ρ^S on transition model $\hat{P}$ for maximizing policy π can be illustrated by:

$$\max_{\pi \in \Pi} \rho^S(\pi) := (1-\lambda) \cdot \underbrace{E\left[\rho(\pi, \hat{P})\right]}_{\text{mean return}} + \lambda \cdot \underbrace{\text{CVaR}^{\alpha}\left[\rho(\pi, \hat{P})\right]}_{\text{robust return}} \tag{3}$$

Here, the soft-robust criterion explicitly includes the mean return (weighted by $1-\lambda$), $\lambda \in [0, 1]$ and CVaR. Combining both, the overall return becomes sensitive to the tail of the distribution and convex. In our problem, we have generated a base MDP with 10 states representing active case percentages. For each binary actions (open:0 or close:1), the transition switches from one state to the other. The rewards are estimated according to the active case counts. For example, if closed and $< 5\%$ sick, the reward is -0.5. This encourages the actor not to close too early to save economy. Similarly, if $\geq 5\%$ people are sick, 0.0 reward is given. For the case of open, if $\geq 5\%$ are sick, -0.5 reward is given to ensure that the actor does not keep open if more than 5% people are sick. Simiarly, for $< 5\%$, depending on how much the number of cases have increased, the *reward* $\in [0, -0.4]$ is calculated. A simplest version of our dataset (with 2 states and 1 outcome) is demonstrated in Table 1. We have extended it further with 10 states and 10 outcomes considering several scenarios for the Governments. Table 2 and 3 illustrates sample state-action with uncertain transition probabilities and rewards on 5 outcomes, for state 0 and state 9, respectively. This addition adds up solving multiple MDPs with different transition probabilities and rewards. A straightforward way to solve is is to use aforementioned Soft-Robust approach on each of the MDPs simultaneously and come up with the robust policy [2]. This approach uses Mixed Integer Linear Programming (MILP) to compute the optimal deterministic policy. This is applicable for our COVID problem as we expect the policies to remain stable as each action (lockdown or not lockdown) can affect many lives, so random action is risky. We discuss the procedure in the next section.

$$\begin{aligned}
\underset{\substack{\pi \in \{0,1\}^{S \times A},\, b \in R,\\ u \in R_+^{S \times A \times N},\, y \in R_+^N}}{\text{maximize}} \quad & \lambda \cdot \Big(b - \frac{1}{1-\alpha} \sum_{\omega \in \Omega} y(\omega)\Big) + (1-\lambda) \cdot \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \sum_{\omega \in \Omega} u(s,a,\omega) \sum_{s' \in \mathcal{S}} r(s,a,s') \cdot P^{\omega}(s,a,s') \\
\text{subject to} \quad & y(\omega) - b \cdot f_\omega \;\geq\; -\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} u(s,a,\omega) \sum_{s' \in \mathcal{S}} P^{\omega}(s,a,s') \cdot r(s,a,s'), && \omega \in \Omega, \\
& \sum_{a \in \mathcal{A}} u(s,a,\omega) \;=\; \sum_{s' \in \mathcal{S}} \sum_{a' \in \mathcal{A}} \gamma \cdot u(s',a',\omega) \cdot P^{\omega}(s',a',s) + f_\omega \cdot p_0(s), && s \in \mathcal{S},\, \omega \in \Omega, \\
& \sum_{a \in \mathcal{A}} \pi(s,a) \;=\; 1, && s \in \mathcal{S}, \\
& u(s,a,\omega) \;\leq\; f_\omega \cdot \pi(s,a)/(1-\gamma), && s \in \mathcal{S},\, a \in \mathcal{A},\, \omega \in \Omega.
\end{aligned}$$

Figure 1: Overview of SR-MILP approach we applied on *COVID MDP*

.

4 Mixed Integer Linear Program for Soft-Robust Optimization

In this section, we discuss our implementation of SR-MILP [2], followed by an analysis on the corresponding output. At first we discuss the equation mentioned in Figure 1. First of all, we aim to solve the objective (first row of the figure). α, b, and λ are the scalar variables. We compute the other components while forming the MDP objective. For example, $u(s, a, \omega) \in R_+$ represents occupancy frequency over state-action pairs. Also, $r(s, a, s')$ and $P^{\omega}(s, a, s')$

Table 1: Our *COVID MDP* simulated dataset for a certain sample activities

idstatefrom	idaction	idstateto	probability	reward	idoutcome
0	1	0	1	0	0
0	0	0	0.969697	0	0
0	0	1	0.030303	-1	0
1	1	1	0.8571429	0	0
1	0	1	1	0	0
1	1	2	0.1428571	-1	0
2	1	2	0.875	0	0
2	0	2	1	0	0
2	1	3	0.125	-1	0
3	0	3	1	0	0
3	1	3	1	0	0

are the reward sets and transition probabilities for each outcome ω. We implemented four constraints (in next four rows). We note that, γ is a scalar value (discount rate). Also f_ω is the model distribution, which is uniform in our case. We implemented the program in Python[1] and optimized using Gurobi Optimizer [2]. We took help from *CRAAM* [3] codebase to organize our objectives and constraints. The output on our simplistic MDP in Table 1 is illustrated in the below Table 4. From there, we can note that, the optimizer always prefers remaining in state 0 in all scenarios. We observe that this phenomenon occurs as our reward and transition probability design. More specifically, if we introduce more outcomes in the existing state-action pairs, we could expect better diversity from the SR-MILP optimizer. While we introduced multiple outcomes for the state-action pairs as mentioned with a few samples in Table 2 and 3, the policy was predicted as defined in Table 5 (our output) and 6. We can observe that, both implementations provide almost similar output, which is, when more than 5% people are sick, the policy encourages "close" action. The only difference between the existing implementation and ours is on closure on state 3, which we leave for further investigation. We have generated multiple figures to illustrate this in a better way, by including *infection rate, daily infected count, and cumulative infected count* based on the number of ongoing days. Figure 2, 3, and 4 depicts the plots for each simulation run from our implementation's policy output. To compare, we have plotted the similar for the authors' *CRAAM* output based on the identical simulation runs, as illustrated in Figure 5, 6, and 7. Surprisingly, we could see that, our policy resulted in lesser cumulative infection count (last row of all figures) compared to the the authors' [2] implementation. This phenomenon occured because our policy predicted "close" action for state 3 in Table 5, whereas the default one didn't in Table 6. In future, we plan to include a comparison between the policies generated from multi-outcome vs. a single-outcome to verify the robustness of SR-MILP. In summary, we claim that, considering our COVID problem, setting that to solve SR-MILP is impactful as we expect a static and robust solution from set of uncertain rewards and transition probabilities to generate a better policy for ensuring public safety.

References

[1] India's daily covid-19 death toll hits record high. `https://www.wsj.com/livecoverage/covid-2021-04-30`. Accessed: 2021-5-3.

[2] Elita A Lobo, Mohammad Ghavamzadeh, and Marek Petrik. Soft-Robust algorithms for batch reinforcement learning. November 2020.

[3] Wikipedia contributors. COVID-19. `https://en.wikipedia.org/w/index.php?title=COVID-19&oldid=1021198556`, May 2021. Accessed: 2021-5-3.

[1] https://www.python.org
[2] https://www.gurobi.com/
[3] https://gitlab.com/RLsquared/craam2

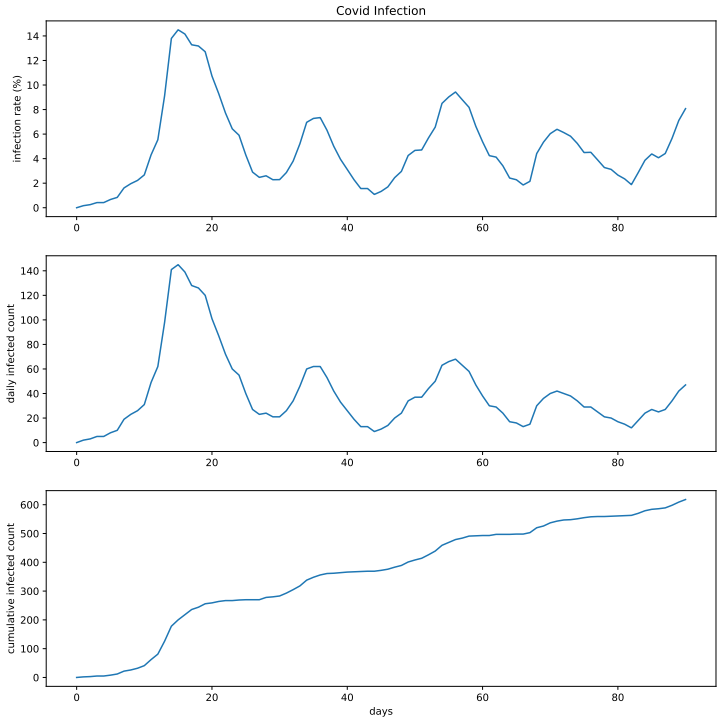

Figure 2: Overall analysis for the policy by our implementation on simulation run 1

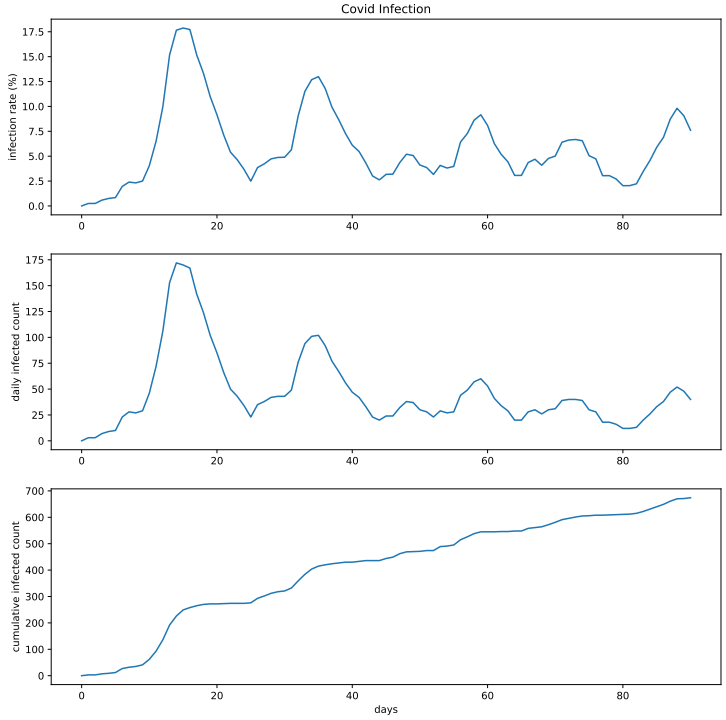

Figure 3: Overall analysis for the policy by our implementation on simulation run 2

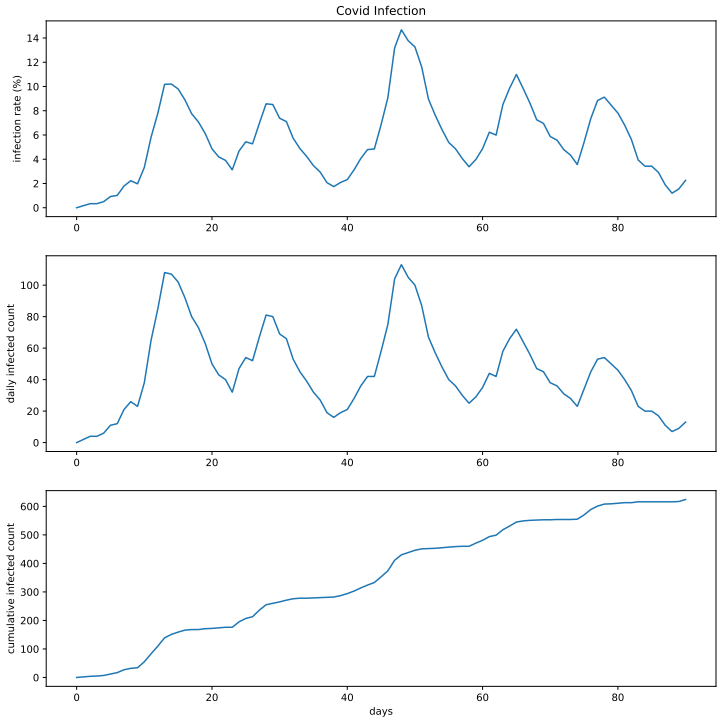

Figure 4: Overall analysis for the policy by our implementation on simulation run 3

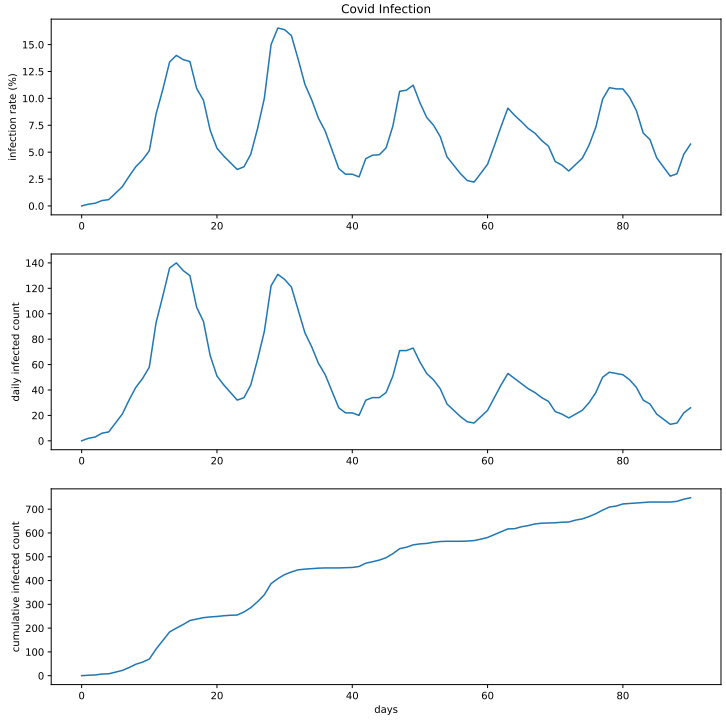

Figure 5: Overall analysis for the policy by authors' *CRAAM* on simulation run 1

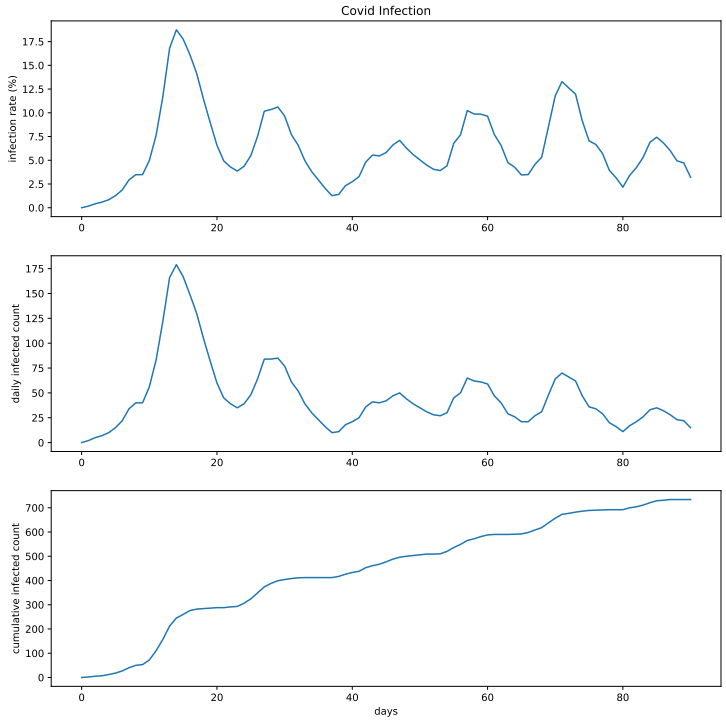

Figure 6: Overall analysis for the policy by authors' *CRAAM* on simulation run 2

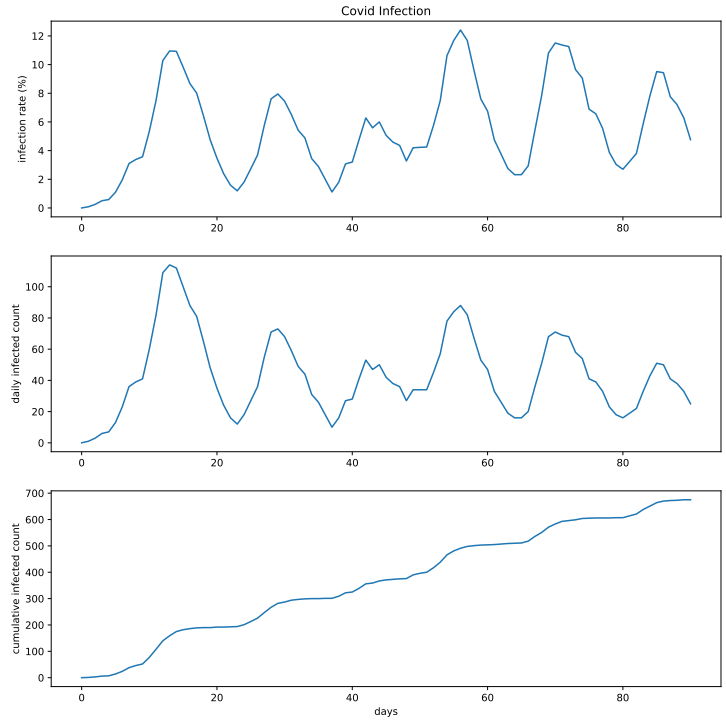

Figure 7: Overall analysis for the policy by authors' *CRAAM* on simulation run 3

Table 2: Generated uncertain probabilities and rewards dataset on 5 outcomes for state 0

idstatefrom	idaction	idstateto	probability	reward	idoutcome
0	0	0	0.5	-0.5	0
0	0	1	0.5	-0.5	0
0	0	0	0.1	-0.5	1
0	0	1	0.1	-0.5	1
0	0	0	1	-0.5	2
0	0	1	0	-0.5	2
0	0	0	1	-0.5	3
0	0	1	0	-0.5	3
0	0	0	0.1	-0.5	4
0	0	1	0.1	-0.5	4
0	0	0	1	-0.5	5
0	0	1	0	-0.5	5

Table 3: Generated uncertain probabilities and rewards dataset on 5 outcomes for state 9

idstatefrom	idaction	idstateto	probability	reward	idoutcome
9	1	8	0.058824	-0.5	0
9	1	9	0.941176	-0.5	0
9	1	8	0	-0.5	1
9	1	9	1	-0.5	1
9	1	8	0	-0.5	2
9	1	9	0.962963	-0.5	2
9	1	8	0	-0.5	3
9	1	9	1	-0.5	3
9	1	8	0	-0.5	4
9	1	9	1	-0.5	4
9	1	8	0	-0.5	5
9	1	9	1	-0.5	5

Table 4: The output from SR-MILP. The risk weight is α, which varies per test run.

risk weight	objective	true return	var	cvar	mean	soft
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0.25	0	0	0	0	0	0
0.25	0	0	0	0	0	0
0.5	0	0	0	0	0	0
0.5	0	0	0	0	0	0
0.75	0	0	0	0	0	0
0.75	0	0	0	0	0	0
1	0	0	0	0	0	0
1	0	0	0	0	0	0

State	Policy
0	OPEN
1	OPEN
2	OPEN
3	CLOSE
4	OPEN
5	CLOSE
6	CLOSE
7	CLOSE
8	CLOSE
9	CLOSE

Table 5: SR-MILP robust policy by our implementation

State	Policy
0	OPEN
1	OPEN
2	OPEN
3	OPEN
4	OPEN
5	CLOSE
6	CLOSE
7	CLOSE
8	CLOSE
9	CLOSE

Table 6: SR-MILP robust policy by the authors' implementation