A Q-Learning Approach to Solve Mixed-Integer Bilevel Linear Programming Problems

Samuel Rodríguez[1,2] and Camilo Gómez[1]

[1]Centro para la Optimización y Probabilidad Aplicada (COPA), Departamento de Ingeniería Industrial, Universidad de los Andes, Bogotá (Colombia)
[2]School of Industrial and Systems Engineering, University of Oklahoma, Norman, OK, USA

Abstract—**In this research project we propose and utilize a novel Reinforcement-Learning-based solution approach to solve Mixed-Integer Bilevel Linear Programming Problems. The use of MIBPs allows to model interactions between two decision makers, while considering potentially complex combinatorial problems. Our methodology, which we named ϵ-greedy Q-MIBLP, can find bilevel optimal solutions for instances from the literature, as well as bilevel feasible and possibly optimal solutions for self generated instances. We tested our methodology approach over a high-dimension combinatorial MIBLP framed in a Public Private Partnership (PPP) context of an infrastructure operation project. Finally, our methodology can find bilevel feasible solutions for the PPP instance within reasonable computation times, allowing for analysis of trade-offs in these kind of scenarios and potential strategies to overcome drawbacks arising from conflicts of interests from the parties involved in these type of associations.**

Keywords: Bilevel Programming, Reinforcement Learning, Q-Learning, Public Private Partnerships, Infrastructure Maintenance, Principal-Agent problem, Stackelberg Games.

I. Introduction

A. What is MIBLP

Stackelberg Games are Game Theoretical scenarios where we find a leader (the principal) that makes decisions pursuing a certain objective and then, sequentially, a follower (the agent) counterpart makes decisions reacting to those of the leader [13]. Bilevel optimization allows to model such situations by incorporating constraints in the follower problem that depend on the leader's decision variables, as well as to model the influence of the follower decisions in the leader's objective function. Bilevel optimization is then a suitable framework for modeling hierarchical two-decision-maker scenarios since it allows to have both 'players' responding to a shared set of rules, but recognizing that each can act in pursuit of their own objectives.

B. Difficulties within MIBLPs

Due to the intrinsic nature of Bilevel Linear Programs (BLPs) where convexity does not hold for the problem's feasible region, these types of problems are classified as NP-Complete [10]. There has been a lot of work devoted to solve BLPs to optimality using single level reformulations that exploit the KKT optimality conditions of the follower sub-problem when all variables are continuous in their domain [21]. However, when integer constraints are enforced to some (if not all) variables in the BLPs, Mixed-Integer Linear Bilevel Programming problems (MIBLPs) arise, significantly increasing the difficulty to optimally solve them.

MIBLPs have drawn the attention of many researchers and academics for their usefulness to model game theoretical combinatorial decision-making processes that consider multiple decision-makers with (possibly) conflicting or diverging objective functions. A wide variety of solution strategies have been proposed to solve MIBLPs to optimality or to retrieve a good quality solution. Some of these approaches include single level reductions applying Branch & Bound methods [4]; descent methods, [9]; Branch & Cut techniques incorporating the ideas of the Corner Polyhedron and Intersection Cuts (ICs) [[7], [5]], among others.

C. Why our approach contributes to the OR community

Nonetheless, to our current knowledge, these solution approaches have been framed in the classic linear programming paradigm and there has not been much assistance from other fields (e.g., the Operations Research or Analytics community) in the search for feasible and possibly optimal solutions to these problems, despite their effectiveness to solve problems within their own domain. For instance, in the field of Machine Learning, supervised learning has shown its effectiveness on prediction problems given a training data set, whereas unsupervised learning has shown its potential to arrange or "cluster" data points that share common characteristics without prior knowledge of the group's labels or even the number of groups in which the data points can be distributed [1]. Supervised and unsupervised learning have been considered the main pillars in the Machine Learning community, but a third pillar has started to take off rapidly: Reinforcement Learning. Reinforcement Learning is a field of Machine Learning that is not considered to be either supervised or unsupervised but an approach to learn, from trial and error, the best action policies for a decision-making process to optimize a given metric [[16], [19]] and to find the best action-policy. The range of problems where Reinforcement Learning has been used is wide and varied, going from robotics [8] to self-driving cars [17], as well as training an automated artificial intelligence that learns to play Atari-like video games in a way that outperforms the most skilled human players [14]. Recent reinforcement learning approaches have been used for to solve management and risk assessment problems in contexts of critical infrastructure systems prone to suffer disruptions on their operations by the occurrence of low-probability extreme events [[12], [2], [23], [6]]. Just as in the MIBLPs framework, where we have not found evidence of other "out-of-the-field" methodologies to solve the aforementioned two-level problems, we have not seen evidence from Machine Learning, specifically the Reinforcement Learning community, in an effort to aid or assist the MIBLPs research community to derive alternative solution techniques.

D. Paper's proposal

In this paper, we focus in one Reinforcement Learning algorithm, named Q-Learning, in order to solve MIBLPs to optimality for a set of instances from the literature, as well as to provide a list of feasible and possibly optimal solutions for a set of self-generated bilevel instances. Additionally, we apply our methodology to a Public Private Partnership case of study addressed in **CITEMYSELF**, where the interactions of a public contractor and a private party are modeled as a MIBLP in the context of an infrastructure operation project. Performance of the learning processes of our implementations are presented as well as the hyper parameters values of choice.

E. Structure

This document is organized as follows. Section **II** presents the literature review related to the problem's context; We review Bilevel Linear Programming problems (BLPs), MIBLPs, and Q-Learning. Section **III** presents, for the very first time, a novel Q-Learning MIBLP solution approach. Section **IV** presents the results for a variety of testbed instances from the literature that we considered in our computational study as well as a set of self-generated instances. Section **V** reviews the PPP problem from the literature and presents its results under our own methodology. Section **VI** synthesizes the main contents of our research; discusses the challenges and our solution approach, while it presents a brief analysis for the overall results of our instances testbed. Moreover, limitations, assumptions, and future work regarding our implementation is encouraged. Finally, the appendix presents the mathematical formulation for every testbed instance, both from the literature and self-generated ones.

II. Literature Review

A. Mixed-Integer Bilevel Linear Programming

B. MIBLPs review

MIBLPs are nested optimization problems where the outer problem is often called the *upper level* or leader problem while the inner problem is called the *lower level* or follower problem. An example of a MIBLP structure, as shown in [7], is presented as follows:

$$\min_{x,y} c_x^T x + c_y^T y \tag{1}$$

$$G_x x + G_y y \leqslant q \tag{2}$$

$$x_j \text{ integer}, \forall\, j \in J_x \tag{3}$$

$$x^- \leqslant x \leqslant x^+ \tag{4}$$

$$y^- \leqslant y \leqslant y^+ \tag{5}$$

$$\begin{aligned} y \in \arg\min_{y \in \mathbb{R}^{n_2}} \{ & d^T y : Ax + By \leqslant b,\ l \leqslant y \leqslant u, \\ & y_j \text{ integer}, \forall\, j \in J_y \} \end{aligned} \tag{6}$$

$$J_x \subseteq N_x := \{1, \dots, n_1\},\ J_y \subseteq N_y := \{1, \dots, n_2\} \tag{7}$$

Where $x \in \mathbb{R}^{n_1}$, $y \in \mathbb{R}^{n_2}$, while c_x, c_y, G_x, G_y, q, d, A, B, b, l, u, x^-, x^+, y^- and y^+ are given matrices/vectors of appropriate size. Also, sets J_x and J_y represent the indexes for the integer-constrained x (leader) and y (follower) variables, respectively [7]. Expressions (1)-(5) correspond to the leader problem while expression (6) corresponds to the follower problem.

C. Importance of MIBLPs and disclaimer for optimistic position

MIBLPs have been granted special attention in recent years due to their challenging nature, depending on which level of the MIBLP (leader or follower) presents the discrete conditions for the corresponding decision variables. Our research assumes an optimistic position for MIBLPs, meaning that given multiple follower optimal solutions, the leader expects the follower to choose the solution that leads to the best objective function value for the leader. The reader can refer to [18] for a detailed review on BLPs history, solution approaches and the pessimistic position.

D. Q-Learning

E. Q-learning Review

Q-Learning is an off-policy Temporal Difference Reinforcement Learning algorithm developed by [22] in the early 90's in order to solve Markov Decision Processes

(MDPs) to optimality and to find their optimal action-policy. The idea behind Q-Learning is to estimate an optimal action-value function for each state known as the Q-function or Q-table, which consists of the expected returns for taking a determined action on a state, and whose convergence is guaranteed under the assumption that all state-action pairs continue to be updated given an exploration policy [19]. The exploration policy is commonly set as to what is known as an ϵ-greedy policy, meaning that, before choosing an action, a random number is generated and, if this number is greater than ϵ, the action selected is going to be greedy with respect to the highest Q-value for the current state (i.e., the $\arg\max_{\mathbf{a}\in\textbf{actions}} Q(s, \mathbf{a})$). Otherwise, the action selected is randomly sampled with equal probability for all actions in the actions pool. The most basic Q-Learning algorithm under an ϵ-greedy exploration policy is depicted in **Algorithm 1** [19].

F. Algorithm description

The expression on line 7 of the Q-Learning algorithm represents the value of taking action **a** in state s at each iteration. The reward r could be a random variable following a specific distribution but can also be deterministic. The transition dynamics from state s to s' are determined by what is known as a state transition function [16]. Parameter α is known as the learning rate or step-size, which is basically a weighted average of past rewards and the initial estimate for Q. Parameter γ can be interpreted in two different ways:

1) When γ is closer to 1, more importance is given to future rewards, whereas a γ value near 0 gives more importance to immediate rewards.
2) It can be seen as a mathematical convenience to guarantee the convergence in the estimation of the optimal Q-function when the underlying MDP consists of only 1 episode with infinite steps.

III. Methodology

A. Methodology proposal

We describe an alternative way to formulate the MIBLP and some necessary concepts.

First the MIBLP has to be restated to its *value function formulation* ([15], [7]) as follows:

$$\min_{x,y} c_x^T x + c_y^T y \tag{8}$$

$$G_x x + G_y y \leqslant q \tag{9}$$

$$A_x x + B_y y \leqslant b \tag{10}$$

$$l \leqslant y \leqslant u \tag{11}$$

$$x_j \text{ integer}, \ \forall\ j \in J_x \tag{12}$$

$$y_j \text{ integer}, \ \forall\ j \in J_y \tag{13}$$

$$d^T y \leqslant \Phi(x) \tag{14}$$

Where $\Phi(x)$ in expression (14) denotes the *follower value function* for a given $x^* \in \mathbb{R}^{n_1}$ by computing the follower MILP:

$$\begin{aligned}\Phi(x^*) := \min_{y\in\mathbb{R}^{n_2}} \{&d^T y : By \leqslant\ b - Ax^*,\ l \leqslant y \leqslant u,\\ &y_j \text{ integer},\ \forall\ j \in J_y\}\end{aligned} \tag{15}$$

By dropping expression (14) from the model (8)-(14), the above formulation results in a MILP called the *high-point-relaxation* (HPR), whose Linear Programming (LP) relaxation is denoted by $\overline{\text{HPR}}$. The following notation:

$$J_F := \{j \in N_x : A_j \neq 0\} \tag{16}$$

is also proposed in [7] to represent the index set of leader variables x_j appearing in the follower problem. In expression (16), A_j represents the j-th column of matrix A.

Bilevel feasibility is violated for a point (x, y) if it violates expression (14), whereas it is satisfied if it met conditions (8)-(14).

Three assumptions need to be considered to successfully retrieve bilevel (possibly optimal) solutions under our Q-learning approach:

1) The $\overline{\text{HPR}}$ feasible set is bounded; the follower MILP (15) has a finite optimal solution $\hat{y}$ for every feasible HPR $(x^*, \cdot)$ [7].
2) The $\overline{\text{HPR}}$ is still feasible for at least one HPR $(x^*, \cdot)$ and its corresponding best-follower response $\hat{y}$.
3) The leader problem has to contain at least one integer constrained variable: $J_x \neq \varnothing$.

B. Implementation details

For the implementation of our Reinforcement Learning based algorithm, which we named ϵ-greedy Q-MIBLP presented below in **Algorithm 3**, we had to perform a small modification to the traditional Q-Learning algorithm shown in algorithm 2, in order to address the CPU memory downsides of saving a large Q table. For the Q-Learning variant, Our algorithm creates the Q table positions "on demand", that is, the respective $Q(s, \cdot)$ for every state s is only created the first time our agent visits that state. For the ϵ-greedy Q-MIBLP, we transform the leader's decision problem from a MILP into an MDP in order to exploit the Q-Learning methodology for which an agent aims to learn the optimal policy consisting in the best actions to take which follows the optimal "trajectory" leading to the best bilevel (possibly optimal) solution. Under this scheme, the state space consist of every possible combination of

Algorithm 1 ϵ-greedy Q-Learning

```
Input: $\alpha \in (0,1]$, $\epsilon > 0$, $\gamma \in [0,1)$, Number of episodes, Number of steps.
1: Initialize $Q(s,a)$ arbitrarily $\forall$ s $\in S$ (if $s$ is a terminal state, set $Q(s,\cdot) \leftarrow 0$).
2: for $episode := 1$ to $Number\ of\ episodes$ do
3:    Initialize starting state $s$.
4:    for $step := 1$ to $Number\ of\ steps$ do
5:       Select a random action $\mathbf{a} \in \mathbf{actions}$ with probability $\epsilon$. Otherwise, set $\mathbf{a} = \arg\max_{\mathbf{a} \in \mathbf{actions}} Q(s,\mathbf{a})$.
6:       Take action $\mathbf{a}$, observe new state $s'$ and reward $r$.
7:       Perform $Q$-Learning action-value update:
            $Q(s,\mathbf{a}) \leftarrow (1-\alpha)\cdot Q(s,\mathbf{a}) + \alpha \cdot [r + \gamma \cdot \max_{\mathbf{a'} \in \mathbf{actions}} Q(s',\mathbf{a'})]$
8:       $s \leftarrow s'$ until $s$ is terminal
9:    end for
10: end for
```

values of the integer constrained leader variables, and the actions or controls are, for each integer constrained leader variable, to add 1 unit, subtract 1 unit or neither add or subtract units to that variable whilst all other integer constrained leader variables remain unchanged. Before creating the actions space, we compute the upper and lower bounds for each leader variable in order to guarantee that our agent keeps moving inside a leader variable "hypercube" that contains the $\overline{\text{HPR}}$ feasible region. The means to retrieve these bounds was to set each leader variable as the objective function of the HPR for both minimization an maximization senses. As can be deduced, the transitions over the state space given a chosen action, say moving from ξ to ξ' after taking action $\mathbf{a}$, are deterministic as well as the immediate rewards. In our approach, the immediate reward of taking an action $\mathbf{a}$ in a certain state ξ was the value of the objective function of the HPR (if bilevel feasible) after solving it by fixing the leader variables to the values of ξ' while also computing the follower subproblem to retrieve $\hat{y} = \arg\min_y \Phi(x^*)$. If the HPR or the follower were found to be infeasible for a given ξ', the reward was penalized with a big negative number. Additionally, in order to avoid our agent to "stick" to a bilevel feasible solution that is known to not be the incumbent, the rewards for the actions that resulted in staying in that specific state were also penalized in order to encourage the agent to move towards the incumbent and giving him the chance to possibly explore vicinity points on its way to the incumbent. These type of transitions between states are repeated for a number of episodes given a starting point ξ and, for each episode, a given number of steps will be performed, which is the same as the number of actions that will be taken in that episode. For the choosing of each episode's starting point, we had two methods to generate them:

1) If a randomly generated number was greater than a parameter $0 \leqslant \omega \leqslant 1$ and an incumbent solution existed, the starting point was going to be the incumbent solution with probability $0 \leqslant \delta \leqslant 1$ in order to explore its neighborhood. Otherwise, with probability $1 - \delta$ the starting point was going to be any of the already found bilevel feasible solutions sampled with equally probability so their neighborhood was to be explored as well.
2) If a randomly generated number was less than ω or if an incumbent has not yet been found, we randomly generated different objective functions for the HPR, then solved the $\overline{\text{HPR}}$, and finally rounded the solution to the nearest integer point for each integer constrained leader variable since the simplex may return a fractional solution point. For the leader variables, the generated HPR objective function coefficients were randomly sampled from an uniform distribution in the interval $[-c_x^T x,\ c_x^T x]$ whereas, for the follower variables, their generated HPR objective function coefficients were randomly sampled from an uniform distribution in the interval $[d_y^T y,\ c_y^T y]$. The reason behind this process is that we wanted our agent to be able to explore the $\overline{\text{HPR}}$'s feasible region starting from different vertices retrieved from the simplex optimal solutions, but trying to be smart in the generation of objective function coefficients whose gradients would aim as precisely as possible to the inducible region [18].

For initializing the values of the Q-table, we solved the $\overline{\text{HPR}}$ under its original objective function and used the negative of this result for each action in each state. Note that this value is equivalent to the dual bound for the associated HPR MILP and, if the associated $\overline{\text{HPR}}$ is feasible, strong duality guarantees that it does not exist an objective function value better than the dual bound. Initializing every Q- value for each state-action pair aids in the exploration process of not-yet-visited state-action couples given the overestimation of the initial Q-values. For the value of ϵ, we initialized it with a value of 1 and decreased it by a percentage after the termination of each episode until a minimum value was reached. Then, after passing this minimum threshold value, ϵ was fixed to a specific value. This epsilon decay strategy encourages exploration at early episodes and exploitation on later episodes while keeping a small chance to randomly select actions from the action pool at every step. The number of episodes for every instance from the literature as well as the self-generated ones was kept the same at 5,000 episodes, whereas the number of episodes for the PPP instance was fixed at 3,000. For each episode, we established the number of steps, or action to take, equal to the size of the action set for each problem. Since our actions are for each integer constrained leader variable, to add 1 unit, subtract 1 unit or neither add or subtract units to that variable–*ceteris paribus*–the size of the action set for each problem is going to be twice as big as the number of integer constrained leader variables (each variable has its corresponding +1 and -1 movement possibilities) plus the additional action of not changing the value over any of the variables. Finally, in **Algorithm 3**, the only MILP that needs to be solved at each step is the follower in order to compute $\Phi(x^*)$ in line 10. When solving the respective $\overline{\text{HPR}}$'s in line 6 and 14 respectively, we are solving a LP with equality constraints on some (if not all) the variables just to perform a feasibility check with no need to implement integer programming techniques (e.g., branch-and-bound or branch-and-cut).

C. *Hardware and software*

We used **Gurobi Optimizer 9.0.1 in Sublime-Python IDE** to compute the initial positions, check HPR and bilevel feasibility, and retrieve the corresponding objectives for both the leader and the follower at every step. Experiments were run in an **HP DESKTOP-OJKJ4ND** with processor **AMD Ryzen 3 2300U with Radeon Vega Mobile Gfx 2.00 GHz** and **12.00 GB RAM**.

IV. Testbed & Results

A. *Small Instances*

B. *Testbed from the literature*

We tested 12 instance problems from the literature ([3], [11], [7], [20]) whose optimal solutions are presented in **Table I**. On average, time required for solving these problems using ϵ-greedy Q-MIBLP depended on the hyper-parameters of choice but the trend was to find the optimal bilevel solution within the first hundred steps for small-sized problems and first hundred episodes for medium-sized problems (size with respect to the number of integer constrained variables present in the leader).

We also tested 5 self generated instance problems whose best known solutions are presented in **Table II**. We believe the objective function values of these problems, as well as their reported variable values, are optimal since we do not have yet any means to prove it other than the observation that the incumbent stopped changing after a considerable amount of episodes. The only problem whose solution is proven to be optimal from our self-generated instances was problem 15, since we were able to 2D-plot the $\overline{\text{HPR}}$ feasible region and find the optimal solution with exhaustive enumeration over the integer constrained inducible region.

C. *Specifications*

For the performance metrics shown below, we considered a moving average with respect to a 500 episode window over 5,000 episodes for all instance problems. We initialized ϵ as 1 at and decayed it with a factor of 0.999 after each episode until it reached a minimum value of 0.1. After ϵ passed its minimum value, we fixed it to 0.3 in order to preserve some exploration probability. This is most likely the reason for the performance drop in all instances after passing the minimum threshold of 0.1, which happened around episode 1800.

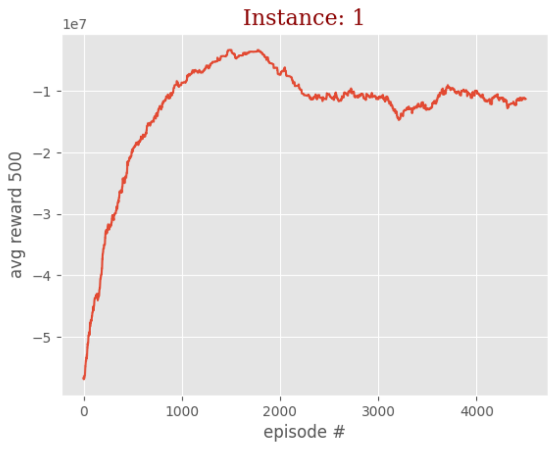

Fig. 1. Learning performance for problem 1

Algorithm 2 ϵ-greedy Q-Learning variant

```
Input: $\alpha \in (0,1]$, $\epsilon > 0$, $\gamma \in [0,1)$, $Number\ of\ episodes$, $Number\ of\ steps$.
 1: for $episode := 1$ to $Number\ of\ episodes$ do
 2:     Initialize starting state $s$.
 3:     if $s$ not in $Q(\cdot, \mathbf{a})$ then
 4:         Initialize $Q(s,\mathbf{a})$ arbitrarily $\forall$ $\mathbf{a} \in$ actions.
 5:     end if
 6:     for $step := 1$ to $Number\ of\ steps$ do
 7:         Select a random action $\mathbf{a} \in$ actions with probability $\epsilon$. Otherwise, set $\mathbf{a} = \arg\max_{\mathbf{a} \in \textbf{actions}} Q(s, \mathbf{a})$.
 8:         Take action $\mathbf{a}$, observe new state $s'$ and reward $r$.
 9:         if $s'$ not in $Q(\cdot, \mathbf{a})$ then
10:             Initialize $Q(s,\mathbf{a})$ arbitrarily $\forall$ $\mathbf{a} \in$ actions.
11:         end if
12:         Perform $Q$-Learning action-value update:
                $$Q(s, \mathbf{a}) \leftarrow (1 - \alpha) \cdot Q(s, \mathbf{a}) + \alpha \cdot [reward + \gamma \cdot \max_{\mathbf{a'} \in \textbf{actions}} Q(s', \mathbf{a'})]$$
13:         $s \leftarrow s'$ until $s$ is terminal
14:     end for
15: end for
```

Algorithm 3 ϵ-greedy Q-MIBLP rewards structure

```
Input: Old state $s$, new state $s'$
Output: The reward of taking a certain action over a certain state.
 1: for $j := 1$ to $\|s'\|$ do
 2:     if $s'_j$ is integer constrained then
 3:         Add constraint $x_j = s'_j$ to $\overline{\text{HPR}}$.
 4:     end if
 5: end for
 6: Solve $\overline{\text{HPR}}$.
 7: if $\overline{\text{HPR}}$ is infeasible then
 8:     reward $\leftarrow -\infty$
 9: else
10:     Solve the follower MILP to compute $\Phi(x^*)$ and the best follower's response $\hat{y}$.
11:     if $\Phi(x^*)$ is not feasible then
12:         reward $\leftarrow -\infty$
13:     else
14:         Build and solve a restricted $\overline{\text{HPR}}$ after fixing all integer contrained variables $x_j = s'_j$ and all $y_j = \hat{y}_j$.
15:         if the restricted $\overline{\text{HPR}}$ is infeasible then
16:             reward $\leftarrow -\infty$
17:         else
18:             reward $\leftarrow -(c_x^T x^* + c_y^T \hat{y})$
19:         end if
20:     end if
21: end if
22: if incumbent exists and $s = s'$ and $s$ is not the incumbent then
23:     reward $\leftarrow -\infty$
24: end if
25: if $c_x^T x^* + c_y^T \hat{y} <$ incumbent then
            $$\text{incumbent} \leftarrow c_x^T x^* + c_y^T \hat{y}$$
26: end if
```

TABLE I
LITERATURE PROBLEMS RESULTS TABLE

Problem	(x^*, y^*)	z_l^*	z_f^*	CPU time (seconds)	Bilevel solutions found
1	(4,4)	-12.00	4.00	43.39	7
2	(6,2)	12.00	-2.00	44-10	35
3	(2,2)	-22.00	2.00	41.34	45
4	(3,1)	5.00	-1.00	42.84	3
5	(2,0,1,0)	-3.50	-4.00	78.02	21
6	(1,9,0)	-19.00	-9.00	43.42	38
7	(0,1,0,1,0,75,21.67)	-1,1011.67	-4,673.33	59.68	108
8	(17,10)	-76.00	30.00	24.49	106
9	(18,14)	-38	14	39.80	84
10	(3,4,2,0,3,0)	49.00	-27.00	94.19	7
11	(0,1,0,1,0,0,1,1,0,1,0,0,0,0)*4	2	-13	314.98	74
12	(1,0,0,1,0,0,1,0)	10	-10	211.09	48

TABLE II
SELF-GENERATED PROBLEMS RESULTS TABLE

Problem	(x^*, y^*)	BKS$_l$	BKS$_f$	CPU time (seconds)	Bilevel solutions found
13	(0,2,0,11,19,0,2,0,0)	-37.00	9.00	228.41	4759
14	(4,27)	-104.00	27.00	32.68	265
15	(1,0,1,1,1)	-3.00	8.00	66.66	1
16	(1,0,0,1.43,0.14)	-0.57	-2.43	41.14	3
17	(3,1,25,2,13,1,5,0,8,7)	1345.00	-148.00	281.24	10693

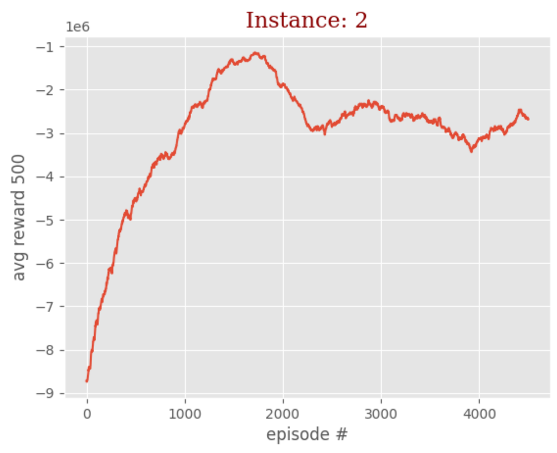

Fig. 2. Learning performance for problem 2

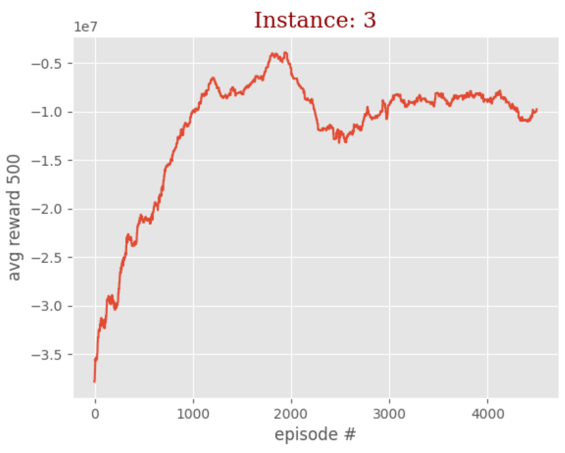

Fig. 3. Learning performance for problem 3

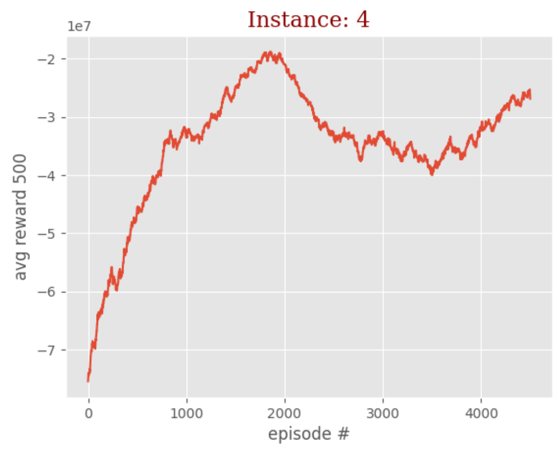

Fig. 4. Learning performance for problem 4

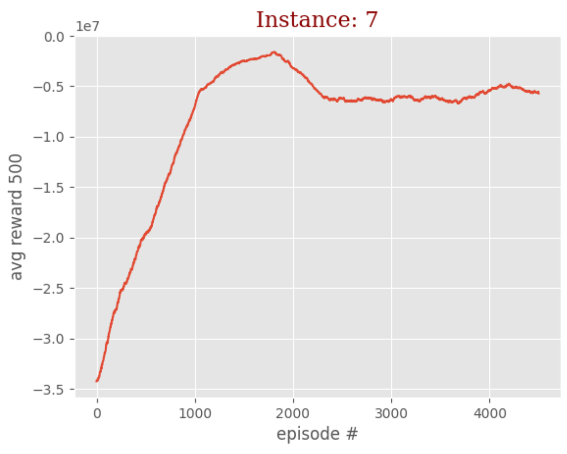

Fig. 7. Learning performance for problem 7

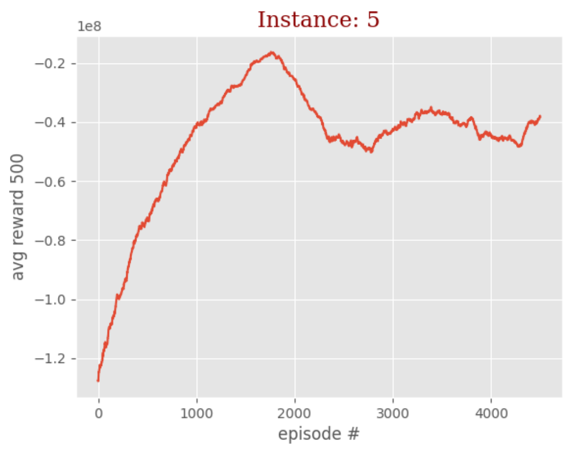

Fig. 5. Learning performance for problem 5

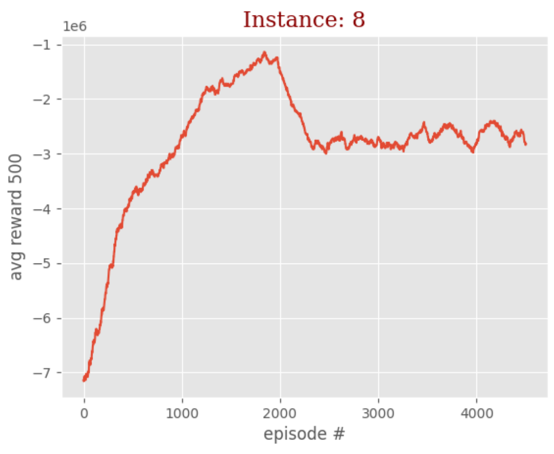

Fig. 8. Learning performance for problem 8

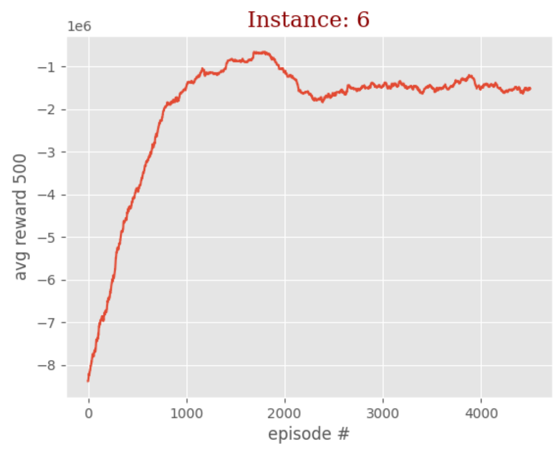

Fig. 6. Learning performance for problem 6

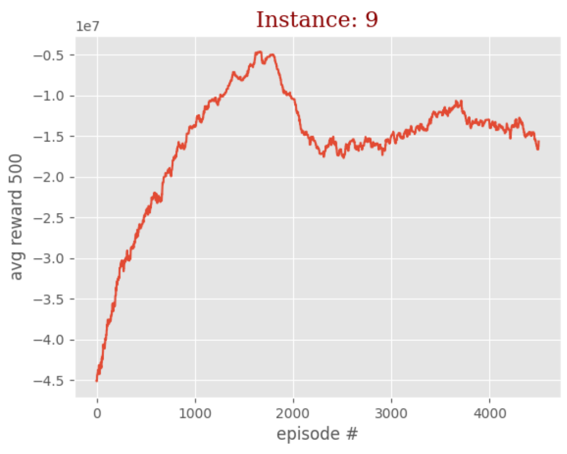

Fig. 9. Learning performance for problem 9

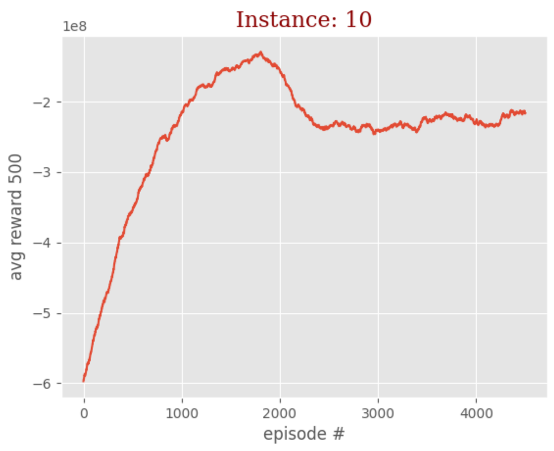

Fig. 10. Learning performance for problem 10

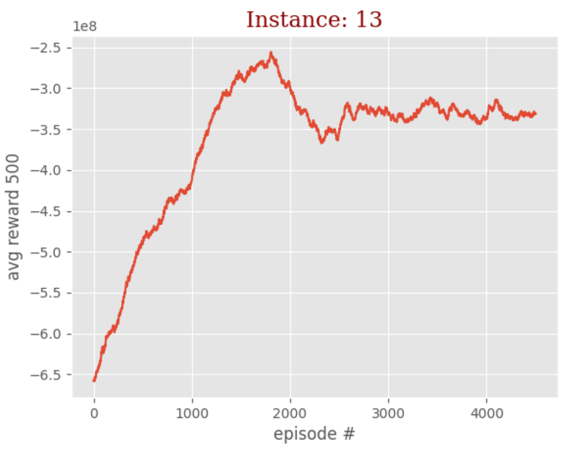

Fig. 13. Learning performance for problem 13

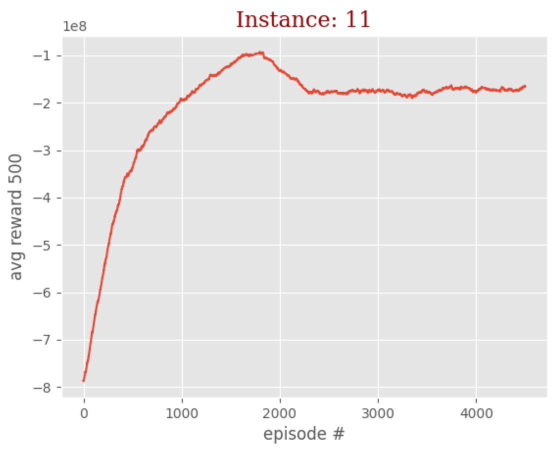

Fig. 11. Learning performance for problem 11

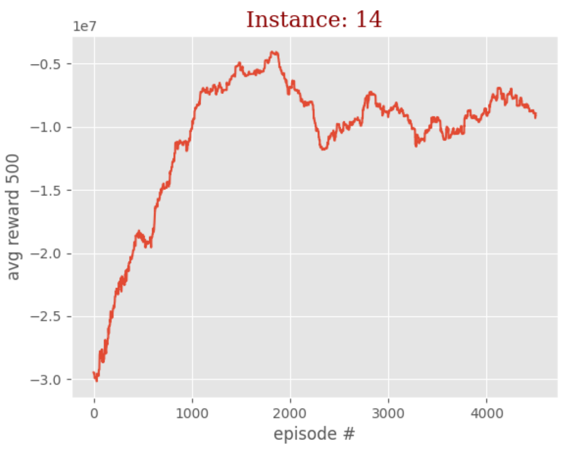

Fig. 14. Learning performance for problem 14

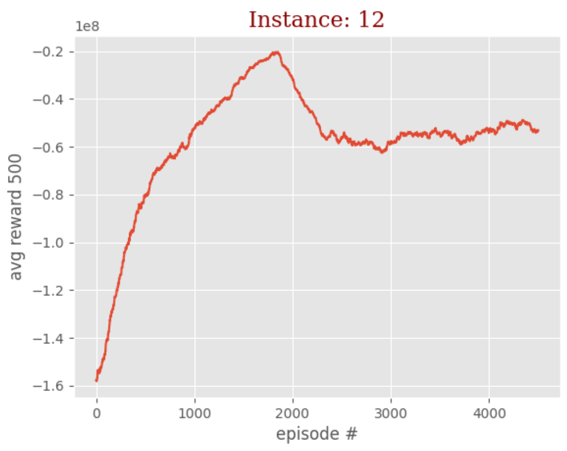

Fig. 12. Learning performance for problem 12

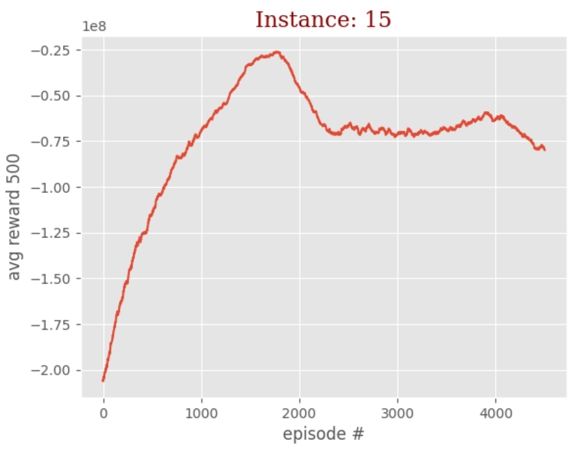

Fig. 15. Learning performance for problem 15

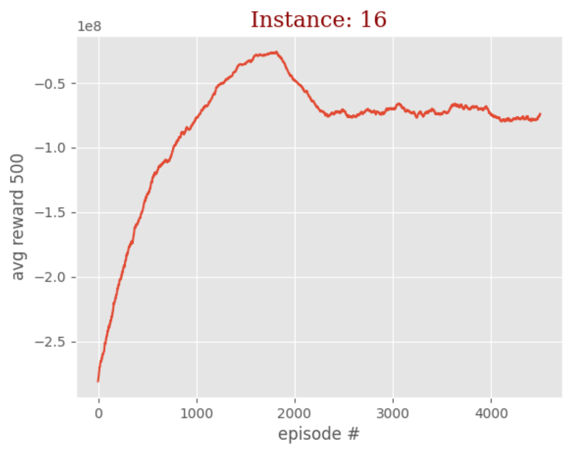

Fig. 16. Learning performance for problem 16

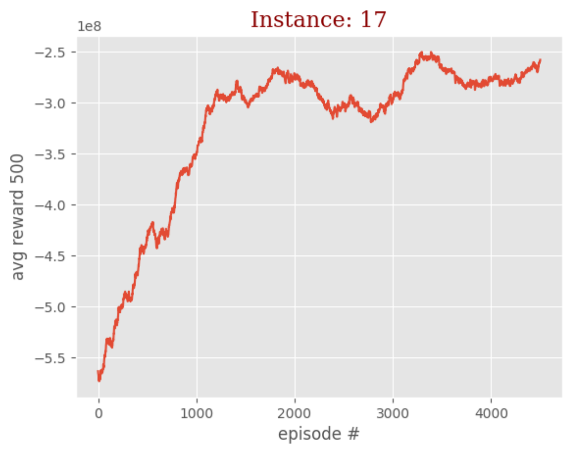

Fig. 17. Learning performance for problem 17

V. PPP Instance

A. Testbed PPP

CYTEMYSELF work consisted on the retrieval and analysis of optimal policies to address the Principal-Agent (PA) problem in PPPs, based on establishing planning actions for a road that suffers performance deterioration over time, while taking into account divergent objectives by the means of a MIBLP formulation. The objective for the principal (the government) objective is to ”guarantee a minimum social benefit derived from the road's performance and service levels throughout a planning horizon”, while the agent ”pursues the cost-effectiveness of the maintenance plan”. In order to solve this problem, a Branch & Cut algorithm was implemented in which, after finding bilevel infeasible solutions in the Branch & Bound tree, a cutting plane named an Intersection Cut (IC)[7] was derived in order to cutoff these solutions. This implementation had a few complications related to the search and retrieval processes of bilevel solutions since it did not have any lower bound to set a stop criterion to the Branch & Bound exploration. Also it is mentioned in **CITEMYSELF** that their algorithm was not being able to find any feasible solutions for different configurations of the parameters set. We implemented our ϵ-greedy Q-MIBLP algorithm over this PPP problem and not only we found better bilevel feasible solutions for the original parameters configuration, but we were also able to find solutions for another configuration for the problem's parameters that were not returning feasible bilevel solutions using the Branch & Cut approach within reasonable computation times.

B. retaking previous work

As mentioned in **CYTEMYSELF**, initial solution for the original parameter's configuration was not bilevel feasible because the minimum social benefit constraint was not satisfied after the private party's maintenance plan responded to the given initial inspection actions from the government. The initial solutions are shown in figures **18 18**.

C. Contribution

It is reasonable to deduce that the government's goal is to find a proper configuration for inspection actions whose private party's maintenance plan response satisfies all the problem's constraints. As mentioned previously in section **III**, the government's (leader) problem is transformed into an MDP in order to find a configuration or combination of integer constrained leader variables, in this case inspection actions, that leads to the best response of the follower variables, in this case the best maintenance plan from the private party that returns a road's performance so the best bilevel feasible social benefit can be attained. The best solution found using the ϵ-greedy Q-MIBLP was

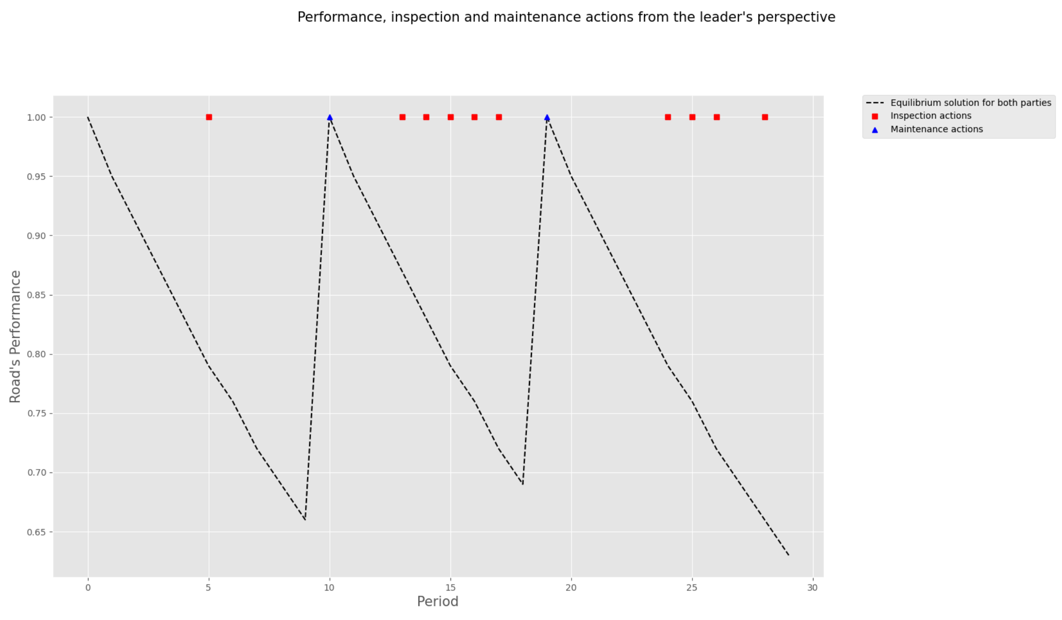

Fig. 18. Solution under the leader's perspective

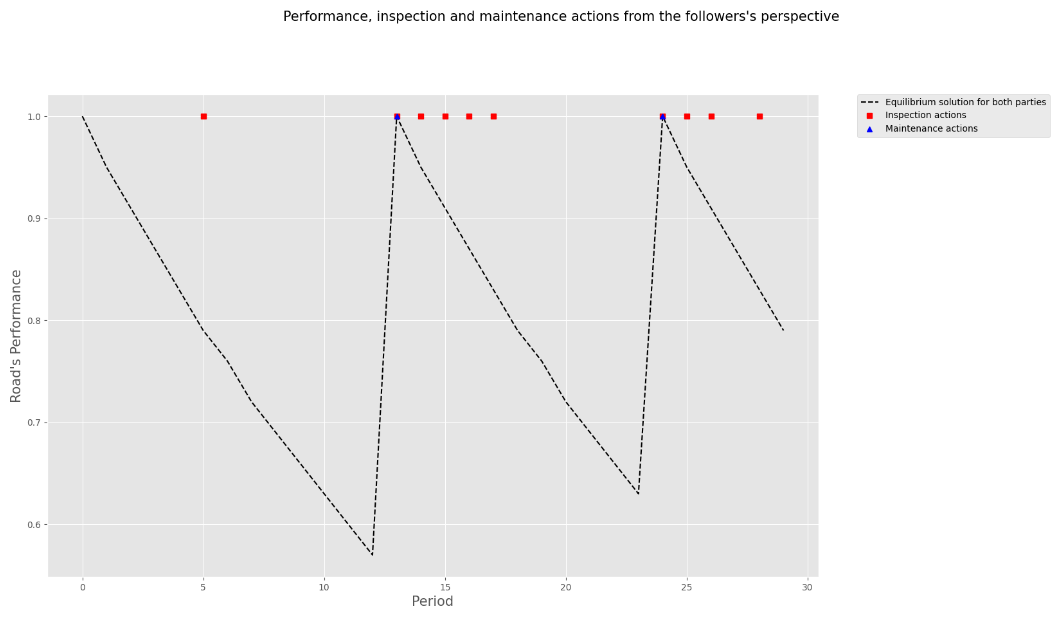

Fig. 19. Solution under the follower's perspective

20% better, in terms of the objective function value, than the one foun by **CITEMYSELF** using the mentioned Branch & Bound approach, being a clear sign of our methodology's outperformance over high-dimension combinatorial problems. Performance of the learning process for the PPP instance is shown in **20**, which consisted of a moving average of the cumulative rewards on a 100 episode window over 3,000 episodes. Also, every time our algorithm found a bilevel solution that updated the incumbent, we saved this solution and, when our algorithm found a solution that had the same objective function value than the incumbent, we saved it as well in order to capture all possible alternate optima. Best bilevel solution is depicted in **Figure 39**.

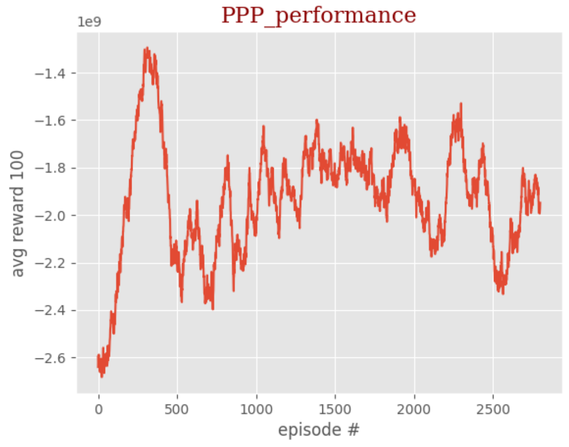

Fig. 20. Learning performance for the PPP instance.

D. Interpreting the results

From the solution figure, it can be seen that our Q-Learning agent is learning which inspection configuration derives in better objective functions for the leader problem by ”turning on/off” inspection actions over the planning horizon and then observing the private party's maintenance response. For the PPP instance, the leader problem consisted of 30 variables and each solution that updated the incumbent is presented in **Table III** alongside their respective leader and follower's objective functions.

For the best solution found, we also saved other 7 bilevel solutions that shared the same objective function, or what is the same, were found to be alternate optima.

E. Disclaimer

The significant drop in performance in **20** is most likely due because the best solution was found on episode 479 and all previous solutions had turn to have a worst reward when visited in order for the agent to always look for the incumbent over the state space. Additionally, the decaying factor for ϵ for the PPP instance was 0.995, so the minimum value 0.1 was passed around episode 460, after which it was fixed to 0.3. Since there is still a lot of variance in the learning process over the average rewards, a rational idea would be to increase the number of episodes. This increase in the number of episodes would give the chance for the agent to visit states that still have an optimistic initial Q-value for certain actions. However, since the incumbent stopped updating itself after around episode 460, we believe this is the optimum for the studied instance and the improvement over the learning process will only improve the action policy in order to reach the incumbent from a given starting point.

F. Contibution to previous work

Finally, As stated in **CITEMYSELF**, It is expected that, when the incentives (from the principal to the agent for having the road at a high service level) increase, fewer inspection actions should be needed in order to guarantee the project's objectives, whereas when the incentives decrease, more inspection actions should be needed since there will not be added value for the agent, in terms of objective functions, if it were to align its interests to the one's of the project. One of the main contributions of this research article is to prove right this reasoning by testing our methodology over a set of incentives whose values were decreased by a small percentage, since this was not possible for the original work in **CITEMYSELF** due to the slow improvements over the bilevel solution retrievals using the Branch & Cut approach. After the solution for this parameters configuration was found, the reasoning was confirmed by observing that the number of inspections performed was greater when the incentives were lower. The initial solutions for this new parameters configuration are shown in figures **22** **22** as well as the best solution found, depicted in **24**, alongside a table showcasing each solution found that sequentially updated the incumbent in **Table IV** with their respective leader and follower's objective functions. Finally, the learning performance for the agent is presented in **??**.

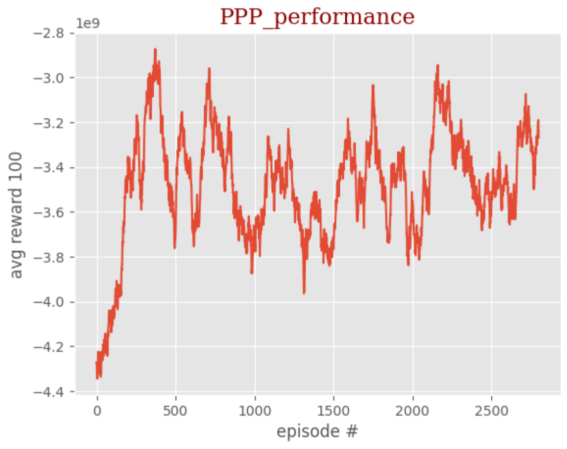

Fig. 25. Learning performance for the PPP instance (second configuration of parameters).

VI. Conclusions

A. Research project summary

We addressed existing MIBLPs instances from the literature, as well as self generated ones, using a novel reinforcement learning approach. We found the optimal solutions and what we believe are the optimal solutions for the latter's and the former's, respectively. We also tested our methodology over a PPP problem modeled as a MIBLP in the context of a road maintenance infrastructure operation project. A key challenge was that the problems

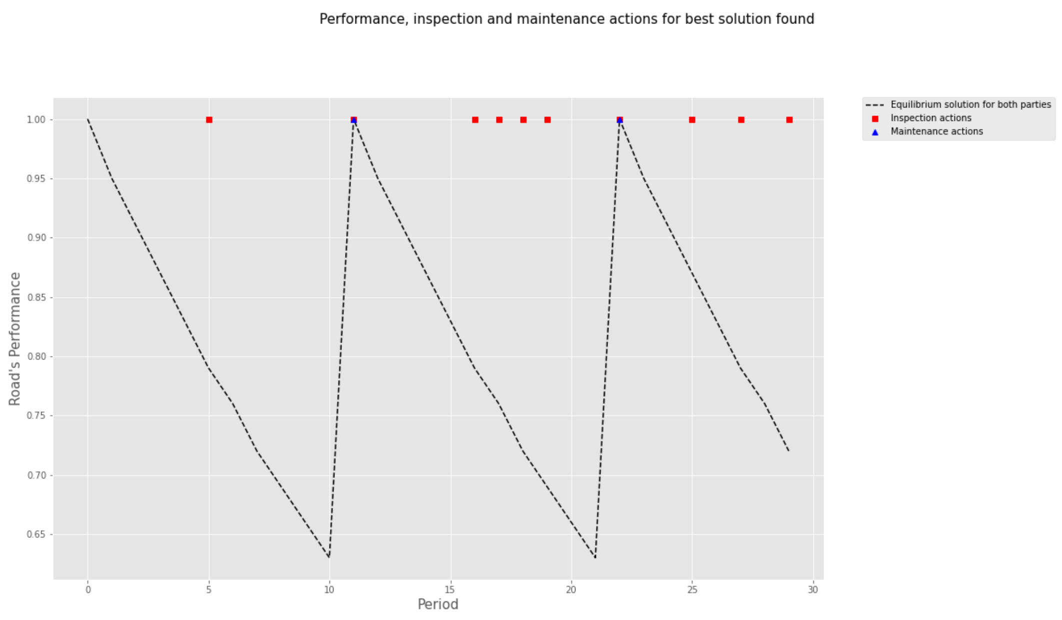

Fig. 21. Best solution found

TABLE III
PPP BILEVEL SOLUTIONS THAT UPDATED THE INCUMBENT

Solution #	Leader O.F.	Follower O.F.
0	-21,763.00	-6,492.00
1	-21,785.50	-6,424.50
2	-21,944.00	-6,171.00
3	-21,977.75	-6,137.25
4	-22,350.00	-5,835.00
5	-23,060.75	-5,264.25
6	-23,433.00	-4,962.00
7	-23,466.75	-4,928.25
8	-23,805.25	-4,659.75
9	-23,839.00	-4,626.00
10	-24,177.50	-4,357.50
11	-24,211.25	-4,323.75
12	-24,245.00	-4,290.00
13	-24,278.75	-4,256.25

TABLE IV
PPP BILEVEL SOLUTIONS THAT UPDATED THE INCUMBENT (SECOND CONFIGURATION OF PARAMETERS)

Solution #	Leader O.F.	Follower O.F.
0	-23,601.00	-4,514.00
1	-23,827.50	-4,357.50
2	-23,850.00	-4,335.00
3	-24,054.00	-4,201.00

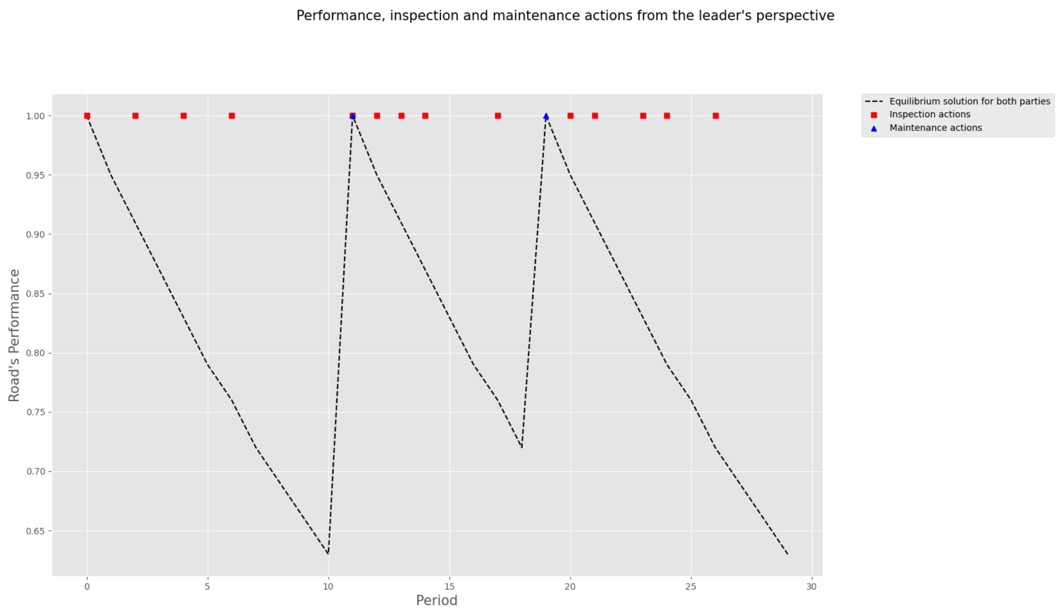

Fig. 22. Solution under the leader's perspective (second configuration of parameters)

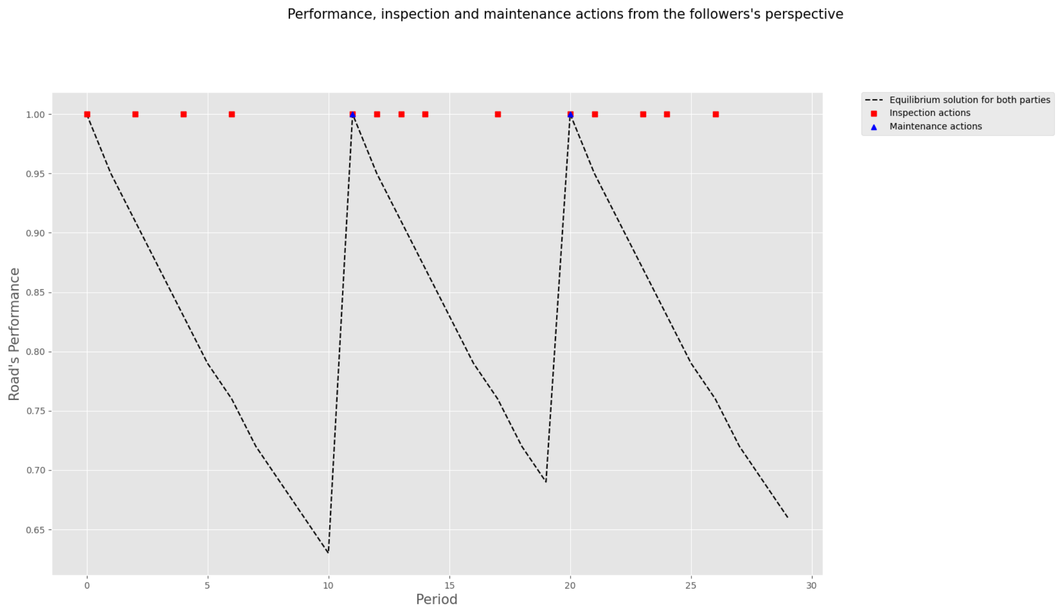

Fig. 23. Solution under the follower's perspective (second configuration of parameters)

under consideration have integer constrained variables, thus, complicating the solution strategies for bilevel approaches. Our methodology outperformed a "state-of-the-art" Branch & Cut approach for solving this particular PPP instance, in which we found a gap of nearly 20% of the solution found using the Branch & Cut compared to our ϵ-greedy Q-MIBLP approach, and our methodology also found solutions for a different configuration of parameters where the latter approach was not able to find them within reasonable computation times. This past couple remarks constitute the main contributions of this research.

B. *Research question*

One important question was addressed in this research:

1) How can machine learning and artificial intelligence assist in the solution process of different operations research methodologies in order to provide alternative ways to address their computational complexity?

To answer this, we proposed a reinforcement learning approach for transforming a MIBLP's leader decision process from the MILP paradigm to an MDP. To our

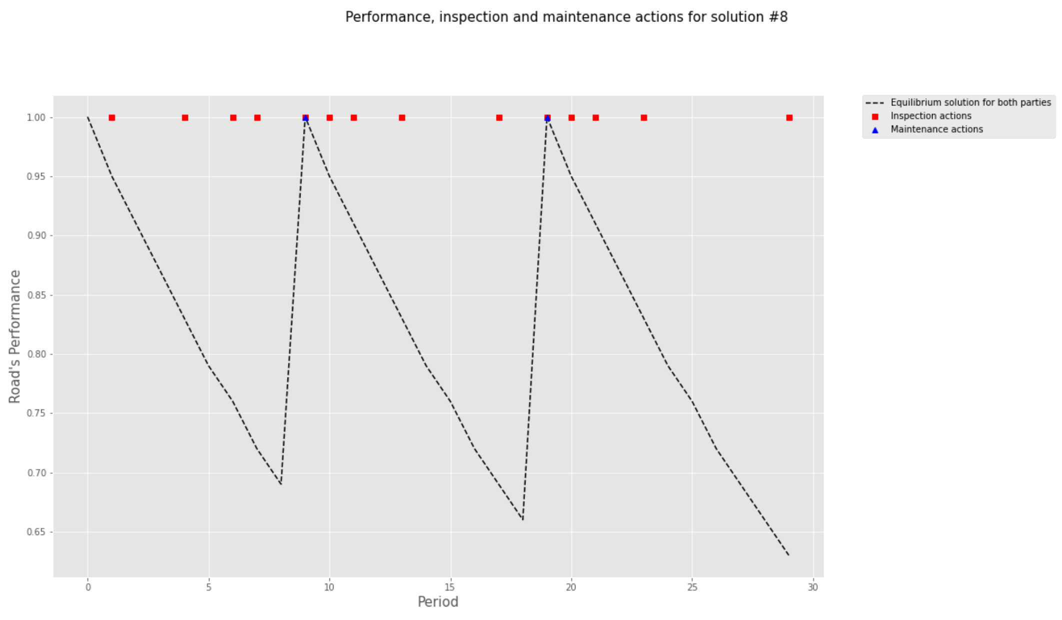

Fig. 24. best solution found (second configuration of parameters)

knowledge, we provided, for the very first time, an algorithm capable of estimating the best leader decisions from a MIBLP using Q-Learning while the rewards were obtained by exploiting the underlying follower MILP after solving it for a given leader solution and retrieving the HPR objective function associated to both the leader and follower decisions at each step.

1) Contribution to the literature: Finally, we provided a PPP contract "simulator" that offers the possibility of making sensitivity analysis over the parameters of interest (e.g., the planning horizon length, the values for the penalties or rewards, the value for the performance obtained after a given number of periods without restoration, among others) that may effect the system in a significant way. The latter with the purpose of finding adequate values for such parameters that estimate the best equilibrium (analogously *bilevel-feasible* solutions) in public-private negotiations before real life implementations.

C. Future Work

Future work is directed towards addressing limitations and relaxing assumptions of our model. On the MIBLP background, we considered the optimistic position for this kind of problems (e.g., given multiple follower optimal solutions, the leader is able to choose the follower's solution that leads to the its best objective function value). We did not test if, for a given leader decision, the follower had multiple responses that leaded to the same value of its objective function, and only considered the possibility of alternate optima for the leader objective. A reasonable extension to our work would be to evaluate if this follower alternate optima situation arises and what would be ways on how to handle them. Also, it is important to highlight that, for both our self-generated instances and the PPP instance, we did not prove optimality by the traditional means of the MILP paradigm since lower bounds for a MIBLP are not well defined as well as we were not using Branch & Bound or Branch & Cut approaches. Under the reinforcement learning paradigm, there is no such thing as optimality proofs but convergence to the optimal policy. Since there is no special interest to find the optimal action policy that takes an agent from any state in the feasible region of a MIBLP problem to the incumbent, we only cared about the incumbent update frequency after the agent has explored a significant amount of the inducible region of the follower problem. When the incumbent stopped updating after a considerable amount of time, we declared it to be the best known solution (possibly optimal) for every instance where there was not any optimal solution found yet (specifically the self generated and PPP instances). Future work would encourage the evaluation of techniques to provide proper lower bounds given a possible relaxation of the HPR, for instance the $\overline{\text{HPR}}$, in order to evaluate how close is every solution found using the ϵ-greedy Q-MIBLP to the lower bound. Additionally, one possible extension for our ϵ-greedy Q-MIBLP would be to incorpotate the usage of Deep Q Networks so an artifical intelligence would be capable of identifying features from the input layer (state space) that are not trivial or that can not be perceived easily by a human modeller. Finally, we encourage the the usage of multi-agent reinforcement learning algorithms to represent the interactions of two decision makers over time, as the case of the PPP addressed hereby and in **CITEMYSELF**.

References

[1] Mohamed Alloghani et al. *A Systematic Review on Supervised and Unsupervised Machine Learning*

Algorithms for Data Science. Ed. by Michael W. Berry, Azlinah Mohamed, and Bee Wah Yap. Cham: Springer International Publishing, 2020, pp. 3–21. ISBN: 978-3-030-22475-2. DOI: 10.1007/978-3-030-22475-2_1. URL: https://doi.org/10.1007/978-3-030-22475-2_1.

[2] C.P. Andriotis and K.G. Papakonstantinou. “Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints”. In: *Reliability Engineering System Safety* 212 (2021), p. 107551. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.107551. URL: https://www.sciencedirect.com/science/article/pii/S095183202100106X.

[3] J. F. Bard. “Some properties of the bilevel programming problem”. In: *Journal of Optimization Theory and Applications* 68.2 (Feb. 1991), pp. 371–378. ISSN: 1573-2878. DOI: 10.1007/BF00941574. URL: https://doi.org/10.1007/BF00941574.

[4] Jonathan F. Bard and James T. Moore. “A Branch and Bound Algorithm for the Bilevel Programming Problem”. In: *SIAM J. Sci. Stat. Comput.* 11.2 (1990), pp. 281–292. ISSN: 0196-5204. DOI: 10.1137/0911017.

[5] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. “Equivalence between intersection cuts and the corner polyhedron”. In: *Oper. Res. Lett.* 38.3 (2010), pp. 153–155. ISSN: 01676377. DOI: 10.1016/j.orl.2010.02.006. URL: www.elsevier.com/locate/orl.

[6] Nariman L. Dehghani, Ashkan B. Jeddi, and Abdollah Shafieezadeh. “Intelligent hurricane resilience enhancement of power distribution systems via deep reinforcement learning”. In: *Applied Energy* 285 (2021), p. 116355. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.116355. URL: https://www.sciencedirect.com/science/article/pii/S0306261920317359.

[7] Matteo Fischetti et al. “A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs”. In: *Oper. Res.* 65.6 (Dec. 2017), pp. 1615–1637. ISSN: 0030-364X. DOI: 10.1287/opre.2017.1650. URL: https://doi.org/10.1287/opre.2017.1650.

[8] Jens Kober, J. Andrew Bagnell, and Jan Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274. DOI: 10.1177/0278364913495721. eprint: https://doi.org/10.1177/0278364913495721. URL: https://doi.org/10.1177/0278364913495721.

[9] C. D. Kolstad and L. S. Lasdon. “Derivative evaluation and computational experience with large bilevel mathematical programs”. In: *J. Optim. Theory Appl.* 65.3 (1990), pp. 485–499. ISSN: 00223239. DOI: 10.1007/BF00939562.

[10] A. Kovács. “On the Computational Complexity of Tariff Optimization for Demand Response Management”. In: *IEEE Transactions on Power Systems* 33.3 (May 2018), pp. 3204–3206. DOI: 10.1109/TPWRS.2018.2802198.

[11] H. Li, L. Zhang, and Y. Jiao. “Solution for integer linear bilevel programming problems using orthogonal genetic algorithm”. In: *Journal of Systems Engineering and Electronics* 25.3 (2014), pp. 443–451.

[12] Yu Liu, Yiming Chen, and Tao Jiang. “Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach”. In: *European Journal of Operational Research* 283.1 (2020), pp. 166–181. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.10.049. URL: https://www.sciencedirect.com/science/article/pii/S0377221719309014.

[13] Ngo Van Long and Gerhard Sorger. “A dynamic principal-agent problem as a feedback Stackelberg differential game”. In: *Central European Journal of Operations Research* 18 (2010), pp. 491–509.

[14] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518 (2015), pp. 529–533.

[15] J. V. Outrata. “On the numerical solution of a class of Stackelberg problems”. In: *Zeitschrift für Operations Research* 34.4 (July 1990), pp. 255–277. ISSN: 1432-5217. DOI: 10.1007/BF01416737. URL: https://doi.org/10.1007/BF01416737.

[16] Warren B. Powell. “A unified framework for stochastic optimization”. In: *European Journal of Operational Research* 275.3 (2019), pp. 795–821. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2018.07.014. URL: https://www.sciencedirect.com/science/article/pii/S0377221718306192.

[17] Florian Pusse and Matthias Klusch. “Hybrid Online POMDP Planning and Deep Reinforcement Learning for Safer Self-Driving Cars”. In: (2019), pp. 1013–1020. DOI: 10.1109/IVS.2019.8814125.

[18] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. “A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications”. In: *IEEE Transactions on Evolutionary Computation* 22 (2017), pp. 276–295.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[20] Yen Tang, Jean-Philippe P. Richard, and J. Cole Smith. “A Class of Algorithms for Mixed-Integer Bilevel Min—Max Optimization”. In: *J. of Global Optimization* 66.2 (Oct. 2016), pp. 225–262. ISSN: 0925-5001. DOI: 10.1007/s10898-015-0274-7. URL: https://doi.org/10.1007/s10898-015-0274-7.

[21] Hoang Tuy, Athanasios Migdalas, and Peter Värbrand. “A global optimization approach for the linear two-level program”. In: *J. Glob. Optim.* 3.1 (1993), pp. 1–23. ISSN: 09255001. DOI: 10.1007/BF01100237. URL: https://link-springer-com.ezproxy.uniandes.edu.co:8443/content/pdf/10.1007%7B%5C%%7D2FBF01100237.pdf.

[22] Christopher Watkins. “Learning From Delayed Rewards”. In: (Jan. 1989).

[23] Zhen-chen Zhou, Zhou Wu, and Tao Jin. “Deep reinforcement learning framework for resilience enhancement of distribution systems under ex-

treme weather events”. In: *International Journal of Electrical Power Energy Systems* 128 (2021), p. 106676. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2020.106676. URL: https://www.sciencedirect.com/science/article/pii/S0142061520342216.

Instance problems:

1)
$$\min x - 4y$$
s.t.,
$$\min y$$
s.t.,
$$-x - y \leqslant -3$$
$$-2x + y \leqslant 1$$
$$2x + y \leqslant 12$$
$$3x - 2y \leqslant 4$$
$$10x + y \geqslant 20$$
$$x,\ y\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

2)
$$\min x + 3y$$
s.t.,
$$x \geqslant 0$$
$$x \leqslant 6$$
$$\min -y$$
s.t.,
$$x + y \leqslant 8$$
$$x + 4y \leqslant 8$$
$$x + 2y \geqslant 13$$
$$x,\ y\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

3)
$$\min -x - 10y$$
s.t.,
$$\min y$$
s.t.,
$$-25x + 20y \leqslant 30$$
$$x + 2y \leqslant 10$$
$$2x - y \leqslant 15$$
$$2x + 10y \geqslant 15$$
$$x,\ y\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

4)
$$\min x + 2y$$
s.t.,
$$\min -y$$
s.t.,
$$-x + 2.5y \leqslant 3.75$$
$$x + 2.5y \geqslant 3.75$$
$$2.5x + y \leqslant 8.75$$
$$x,\ y\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

5)
$$\min -2x_1 + x_2 + 0.5y_1$$
s.t.,
$$x_1 + x_2 \leqslant 2$$
$$\min -4y_1 + y_2$$
s.t.,
$$2x_1 - y_1 + y_2 \geqslant 2.5$$
$$-x_1 + 3x_2 - y_2 \geqslant -2$$
$$x_1,\ x_2,\ y_1,\ y_2\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

6)
$$\min -x - 2y_1 - 3y_2$$
s.t.,
$$0 \leqslant x \leqslant 8$$
$$\min -y_1 + y_2$$
s.t.,
$$x + y_1 + y_2 \leqslant 10$$
$$0 \leqslant y_1 \leqslant 9$$
$$0 \leqslant y_2 \leqslant 7$$
$$x,\ y_1,\ y_2\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

7)
$$\min -20x_1 - 60x_2 - 30x_3 - 50x_4 - 15y_1 - 10y_2 - 7y_3$$
s.t.,
$$\min -20y_1 - 60y_2 - 8y_3$$
s.t.,
$$5x_1 + 10x_2 + 30x_3 + 5x_4 + 8y_1 + 2y_2 + 3y_3 \leqslant 230$$
$$20x_1 + 5x_2 + 10x_3 + 10x_4 + 4y_1 + 3y_2 \leqslant 240$$
$$5x_1 + 5x_2 + 10x_3 + 5x_4 + 2y_1 + y_3 \leqslant 90$$
$$x_1,\ x_2,\ x_3,\ x_4\ \in\ \{0,1\}$$
$$y_1,\ y_2,\ y_3\ \geqslant\ 0$$

8)
$$\min 2x - 11y$$

s.t.,

$$\min x + 3y$$

s.t.,

$$x - 2y \leqslant 4$$
$$2x - y \leqslant 24$$
$$3x + 4y \leqslant 96$$
$$x + 7y \leqslant 126$$
$$-4x + 5y \leqslant 65$$
$$-x - 4y \leqslant -8$$
$$x,\ y\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

9)

$$\min x - 4y$$

s.t.,

$$\min y$$

s.t.,

$$-2x + y \leqslant 0$$
$$2x + 5y \leqslant 108$$
$$2x - 3y \leqslant -4$$
$$x,\ y\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

10)

$$\min -4x_1 + 8x_2 + x_3 - x_4 + 9y_1 - 9y_2$$

s.t.,

$$-9x_1 + 3x_2 - 8x_3 + 3x_4 + 3y_1 \leqslant 1$$
$$4x_1 - 10x_2 + 3x_3 + 5x_4 + 8y_1 + 8y_2 \leqslant 25$$
$$4x_1 - 2x_2 - 2x_3 + 10x_4 - 5y_1 + 8y_2 \leqslant 21$$
$$9x_1 - 9x_2 + 4x_3 - 3x_4 - y_1 - 9y_2 \leqslant -1$$
$$-2x_1 - 2x_2 + 8x_3 - 5x_4 + 5y_1 + 8y_2 \leqslant 20$$
$$7x_1 + 2x_2 - 5x_3 + 4x_4 - 5y_1 \leqslant 11$$
$$\min -9y_1 + 9y_2$$

s.t.,

$$-6x_1 + x_2 + x_3 - 3x_4 - 9y_1 - 7y_2 \leqslant -15$$
$$4x_2 + 5x_3 + 10x_4 \leqslant 26$$
$$-9x_1 + 9x_2 - 9x_3 + 5x_4 - 5y_1 - 4y_2 \leqslant -5$$
$$5x_1 + 3x_2 + x_3 + 9x_4 + y_1 + 5y_2 \leqslant 32$$
$$x_1,\ x_2,\ x_3,\ x_4,\ y_1\ y_2\ \in\ \mathbb{Z}^+\ \cup\ \{0\}$$

11)

$$\min \sum_{i=0}^{6} x_i$$

s.t.,

$$y_6 + x_6 \leqslant 1$$
$$3y_0 + 3y_1 + 3y_2 + 4y_3 + 4y_4 + 5y_5 + 6y_6 \leqslant 13$$
$$\min -3y_0 - 3y_1 - 3y_2 - 4y_3 - 4y_4 - 5y_5 - 6y_6$$

s.t.,

$$y_0 + 2y_1 + 2y_2 + 3y_3 + 3y_4 + 4y_5 + 5y_6 \leqslant 10$$
$$y_i + x_i \leqslant 1\ \forall\ i \in \{0, \ldots, 5\}$$
$$x_i,\ y_i,\ \in\ \mathbb{Z}^+\ \cup\ \{0\}\ \forall\ i \in \{0, \ldots, 6\}$$

12)

$$\min 8x_1 + 7x_2 + 10x_3 + 4x_4$$

s.t.,

$$\sum_{i=1}^{4} w_i \leqslant 2$$
$$\min -8x_1 - 7x_2 - 10x_3 - 4x_4$$

s.t.,

$$3x_1 + 4x_2 + 6x_3 + 3x_4 \leqslant 9$$
$$x_i \leqslant 1 - w_i\ \forall\ i \in \{1, 2, 3, 4\}$$
$$x_i,\ w_i\ \in\ \{0, 1\}\ \forall\ i \in \{1, 2, 3, 4\}$$

13)

$$\min -2x_1 + x_2 + x_3 - 2x_4 - x_5 + 3.5x_6$$
$$+y_1 + 1.5y_2 - 3y_3$$

s.t.,

$$\min -2x_2 + x_5 - 3y_1 + y_2 + 4y_3$$

s.t.,

$$-x_1 + 0.2x_2 + x_5 + 2x_6 - 4y_1 + 2y_2 + y_3 \leqslant 12$$
$$x1 + x_3 - 2x_4 - 4y_2 + y_3 \leqslant 10$$
$$5x_1 + x_4 + 3.2x_6 + 2y_1 + 2y_2 \leqslant 15$$
$$-3 * x_2 - x_4 + x_5 - 2y_1 \leqslant 12$$
$$-2x_1 - x_2 - y_2 + y_3 \leqslant -2$$
$$-y_1 - 2y_2 - y_3 \leqslant -2$$
$$-2x_2 - 3x_3 - x_5 \leqslant -3$$
$$x_i \in\ \mathbb{Z}^+\ \cup\ \{0\}\ \forall\ i \in \{1, \ldots, 6\}$$
$$y_i \in\ \mathbb{Z}^+\ \cup\ \{0\}\ \forall\ i \in \{1, \ldots, 3\}$$

14)

$$\min x - 4y$$

s.t.,

$$\min y$$

s.t.,

$$-\frac{12}{17}x + \frac{5}{6}y \leqslant 26$$
$$-\frac{5}{18}x + y \leqslant 37$$
$$-\frac{4}{19}x + \frac{4}{3}y \leqslant 55$$

$$-\frac{2}{6}x - y \leqslant -17.5$$
$$-x \leqslant -3.5$$
$$-\frac{8}{9}x - \frac{7}{30}y \leqslant -9.8$$
$$-\frac{12}{17}x - y \leqslant -22$$
$$-2x - 10y \leqslant -147$$
$$-y \leqslant -8.8$$
$$2x - y \leqslant 110$$
$$x + 5y \leqslant 303$$
$$\frac{17}{31}x + \frac{1}{10}y \leqslant 41.7$$
$$2x + 2y \leqslant 225$$
$$x - 4y \leqslant 6$$
$$x \leqslant 69$$
$$x - 2y \leqslant 30$$
$$y \leqslant 48.8$$
$$-\frac{7}{13}x + y \leqslant 33.2$$
$$-x - \frac{1}{2}y \leqslant -16$$
$$\frac{1}{4}x - \frac{1}{4}y \leqslant 11.2$$
$$x, y \in \mathbb{Z}^+ \cup \{0\}$$

15)
$$\min 5x_1 + 7x_2 - 10y_1 + 3y_3 - 1y_3$$
s.t.,
$$\min 10y_1 - 3y_3 + 1y_3$$
s.t.,
$$-x_1 + 3x_2 - 5y_1 - y_2 + 4y_3 \leqslant -2$$
$$-2x_1 + 6x_2 - 3y_1 - 2y_2 + 2y_3 \leqslant -5$$
$$x2 - 2y_1 + y_2 - y_3 \leqslant -2$$
$$x_i, \in \{0,1\} \; \forall \; i \in \{1,2\}$$
$$y_i, \in \{0,1\} \; \forall \; i \in \{1,2,3\}$$

16)
$$\min -3x_1 - 2x_2 + 5x_3 + 2y_1 - 3y_2$$
s.t.,
$$\min -2y_1 + 3y_2$$
s.t.,
$$x_1 + x_2 + x_3 + 2y_1 + y_2 \leqslant 4$$
$$7x_1 + 3x_3 - 4y_1 + 3y_2 \leqslant 7$$
$$-11x_1 + 6x_2 + y_1 - 3y_2 \leqslant -10$$
$$x_i, \in \{0,1\} \; \forall \; i \in \{1,2,3\}$$
$$y_i, \geqslant 0 \; \forall \; i \in \{1,2\}$$

17)
$$\min 25x_1 - 1.5x_2 - 10x_3 - 0.5x_4 - 3x_5 - 2.5x_6 + 87y_1 + 101y_2 + 93y_3 + 55y_4$$
s.t.,
$$\min -3y_1 - 5y_2 - 7y_3 - 11y_4$$
s.t.,
$$2x_1 + 3x_2 + 7x_3 + 6x_4 + x_5 + 3x_6 + 5y_1 + 5y_2 - 9y_3 + 4y_4 \leqslant 193$$
$$x_1 + 4x_2 + 7x_3 + 5x_4 + 7x_6 - 4y_1 + y_3 - 2y_4 \leqslant 173$$
$$-x_1 - 2x_2 - x_3 - x_4 - 3x_5 - 5x_6 - 7y_1 - 3y_2 - 4y_3 - 7y_4 \leqslant -31$$
$$4x_1 + 3x_2 + 7x_3 + 7x_5 + 7x_6 + 5y_1 - 3y_2 + 2y_3 - 3y_4 \leqslant 310$$
$$-4x_1 - x_2 - 3x_3 - 5x_4 - 3x_5 - 7x_6 - 3y_1 - 2y_2 - 5y_3 - 6y_4 \leqslant -214$$
$$2x_1 + 7x_2 + 3x_3 + 4x_4 + 8x_5 + 8x_6 - 6y_1 + y_2 - 4y_3 + y_4 \leqslant 157$$
$$y_1 \leqslant 23$$
$$y_2 \leqslant 4$$
$$y_3 \leqslant 10$$
$$y_4 \leqslant 7$$
$$x_i \in \mathbb{Z}^+ \cup \{0\} \; \forall \; i \in \{1, \ldots, 6\}$$
$$y_i \in \mathbb{Z}^+ \cup \{0\} \; \forall \; i \in \{1, \ldots, 4\}$$

PPP solutions found:

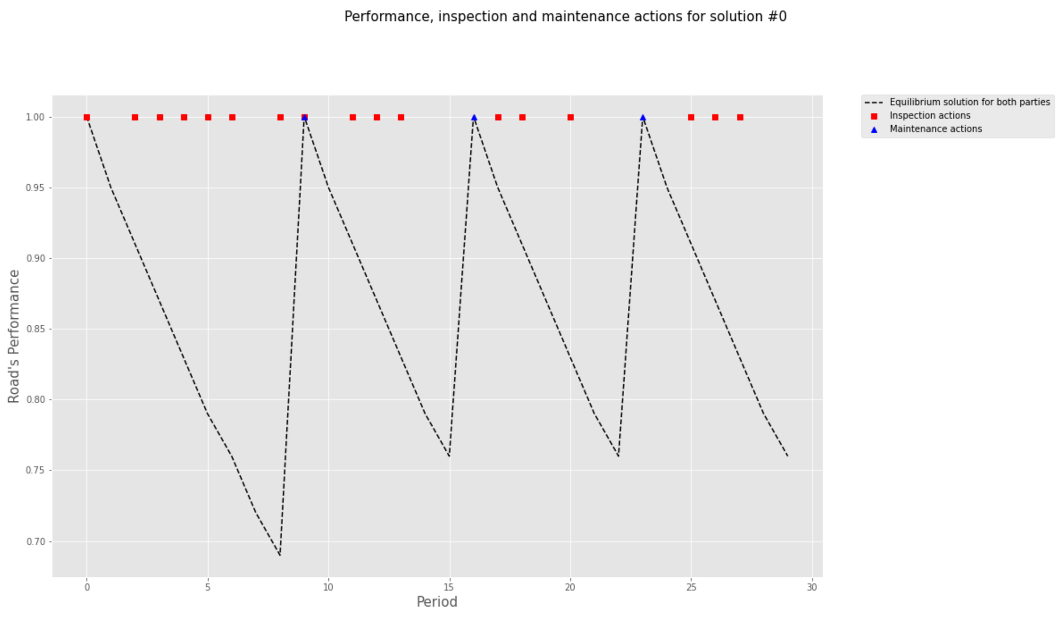

Fig. 26. Solution 0

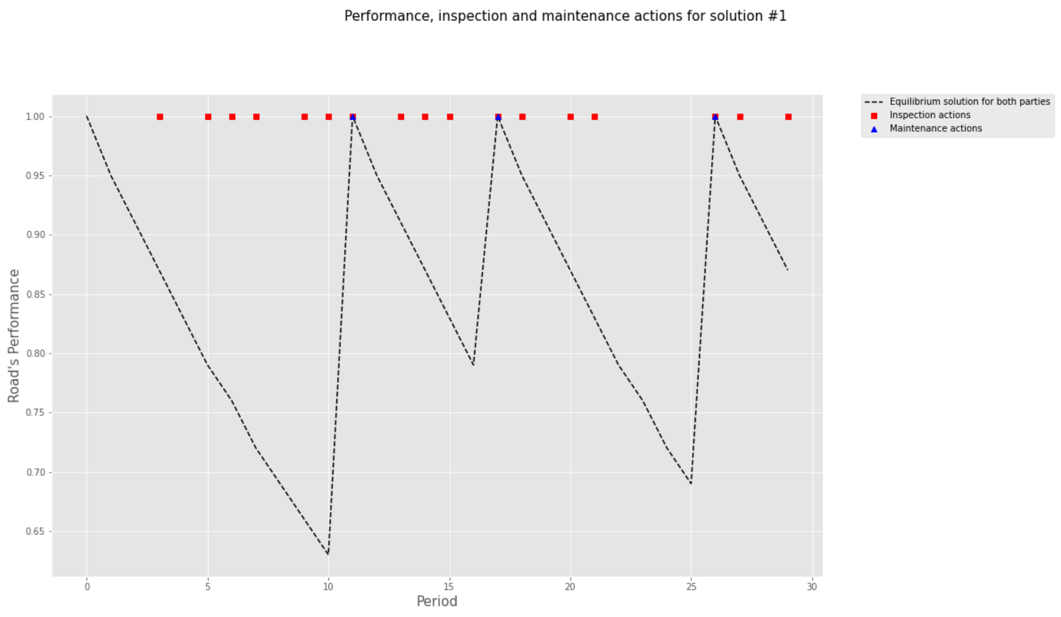

Fig. 27. Solution 1

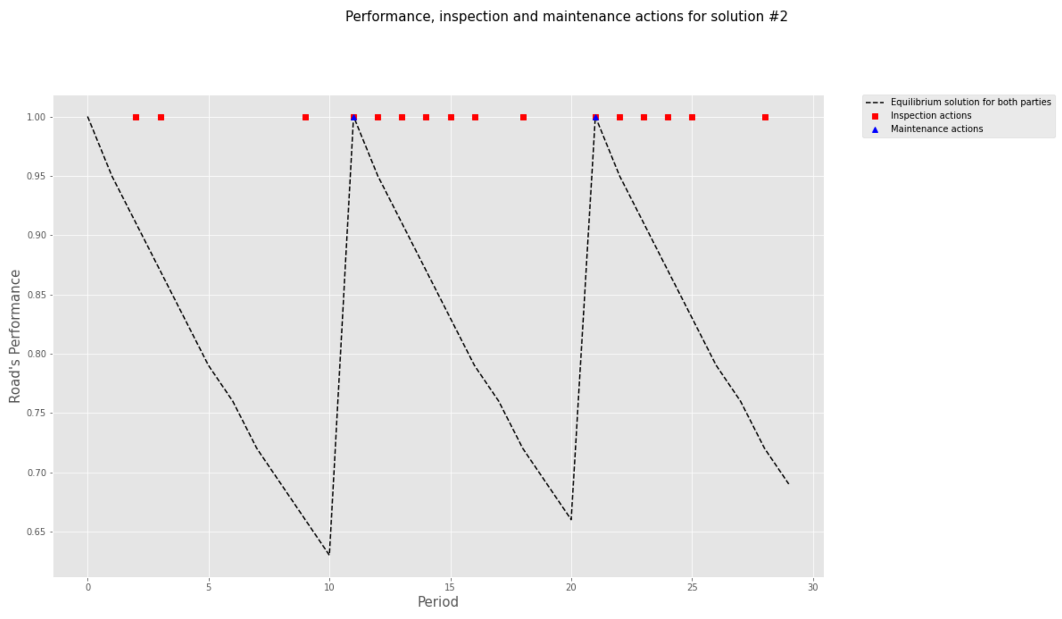

Fig. 28. Solution 2

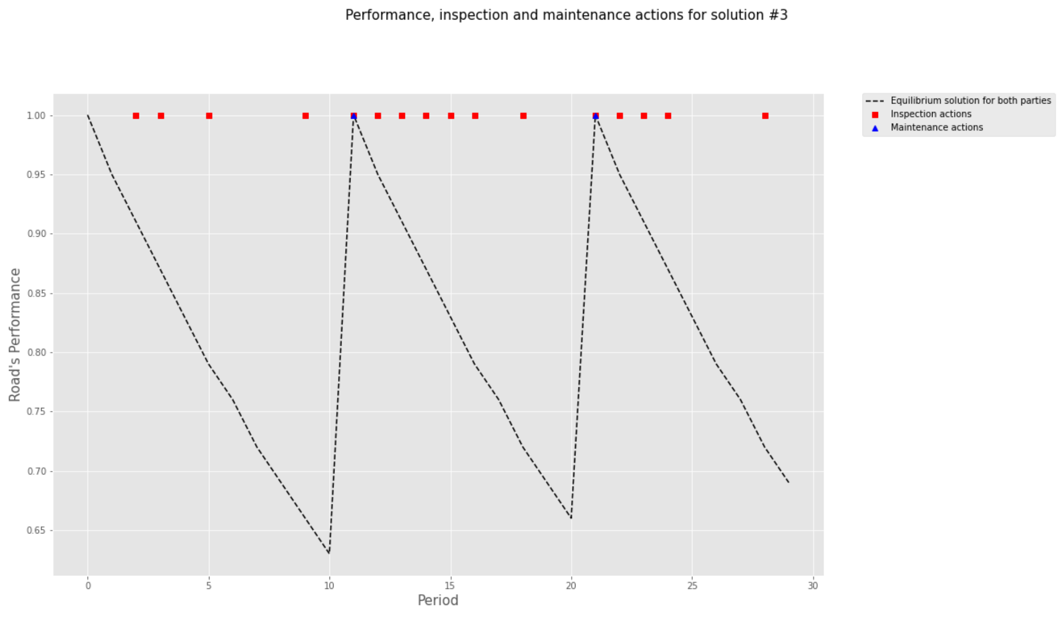

Fig. 29. Solution 3

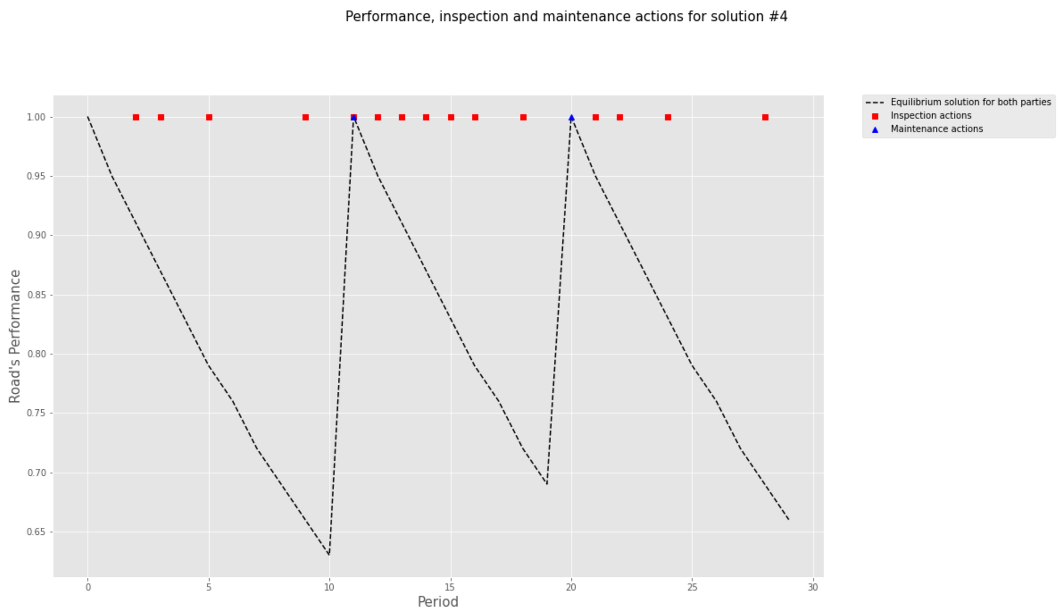

Fig. 30. Solution 4

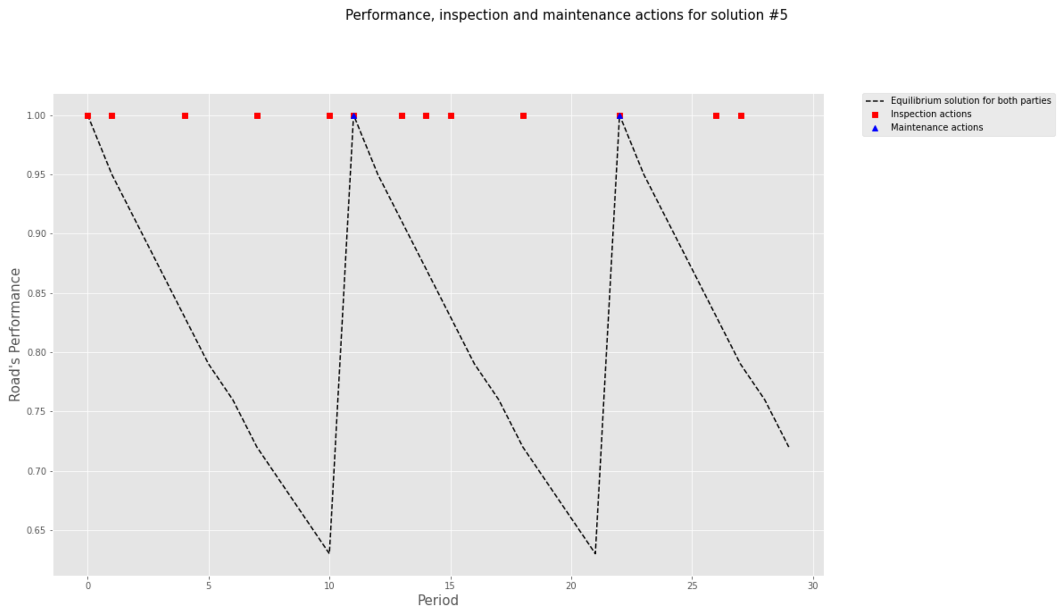

Fig. 31. Solution 5

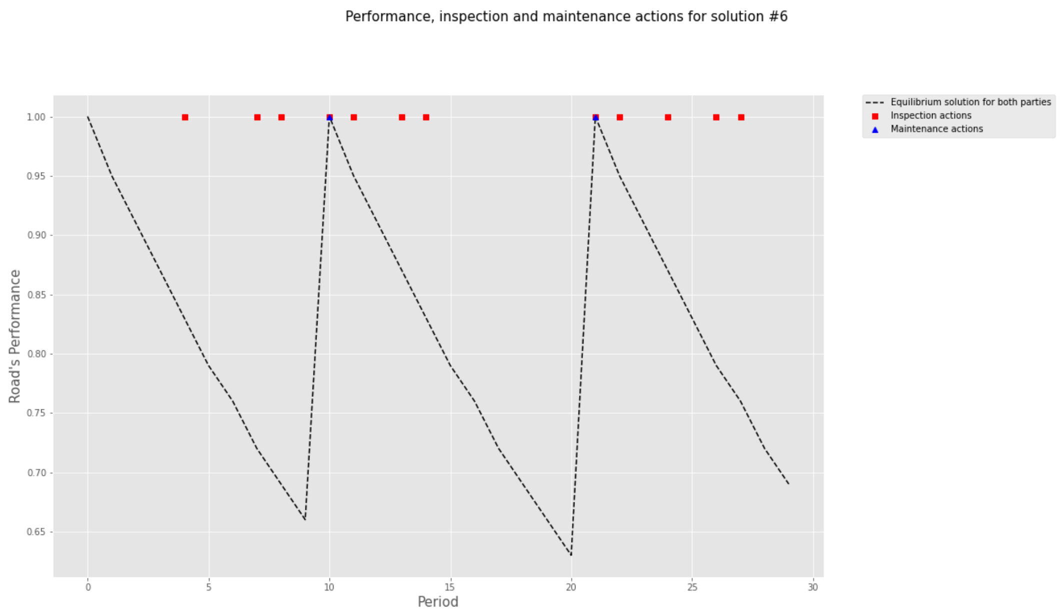

Fig. 32. Solution 6

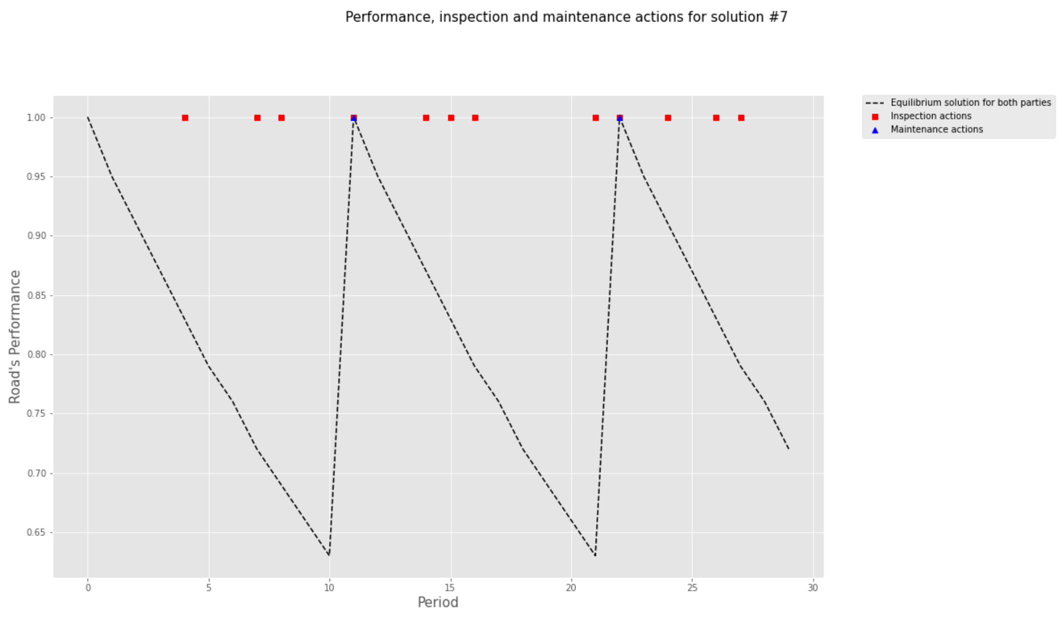

Fig. 33. Solution 7

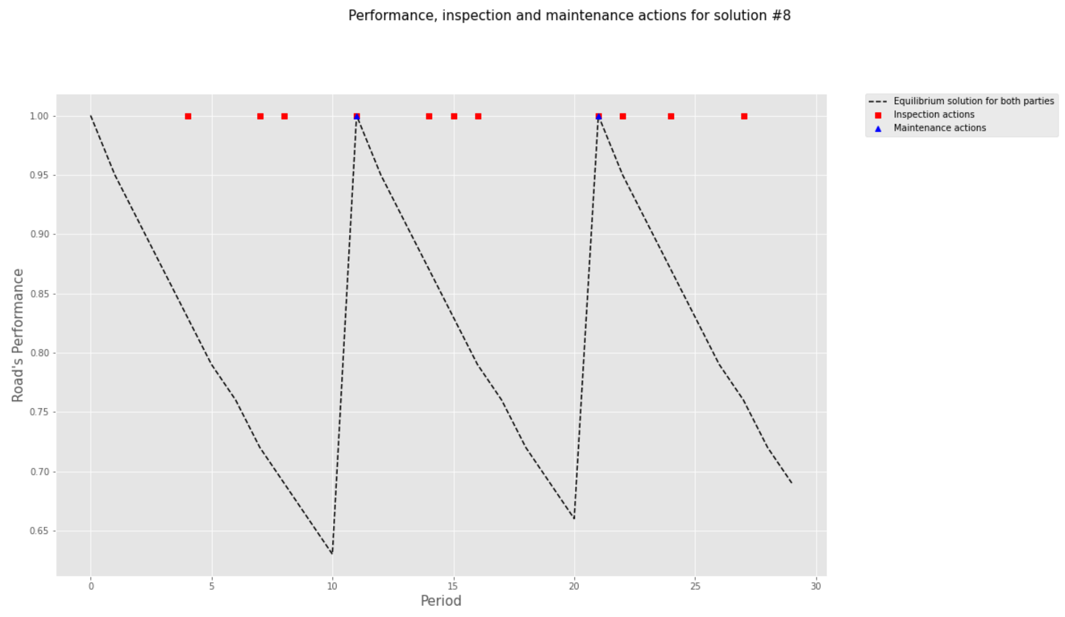

Fig. 34. Solution 8

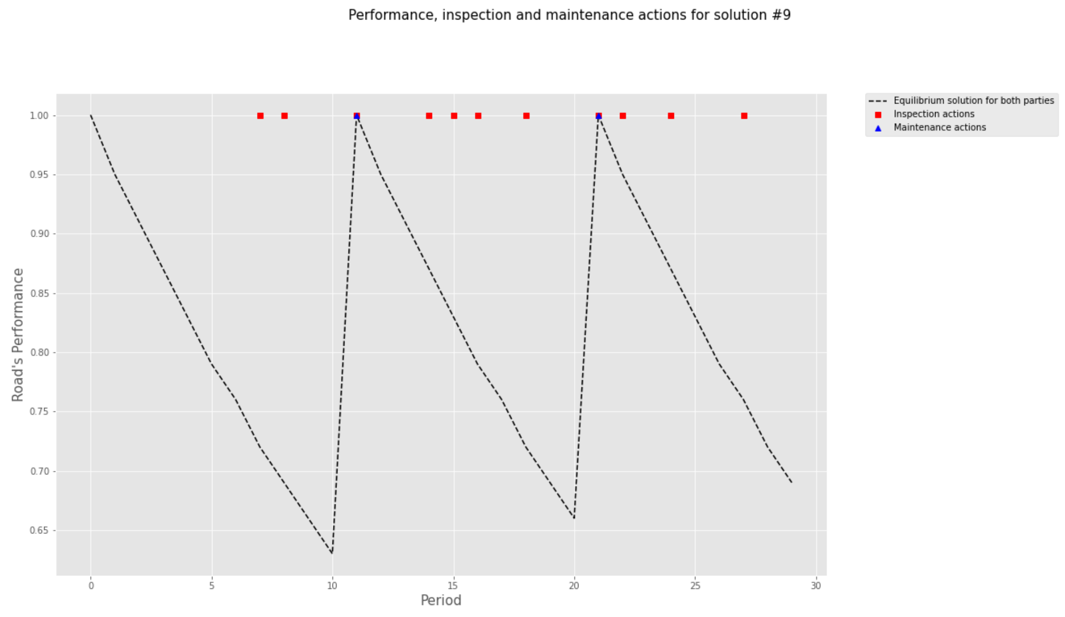

Fig. 35. Solution 9

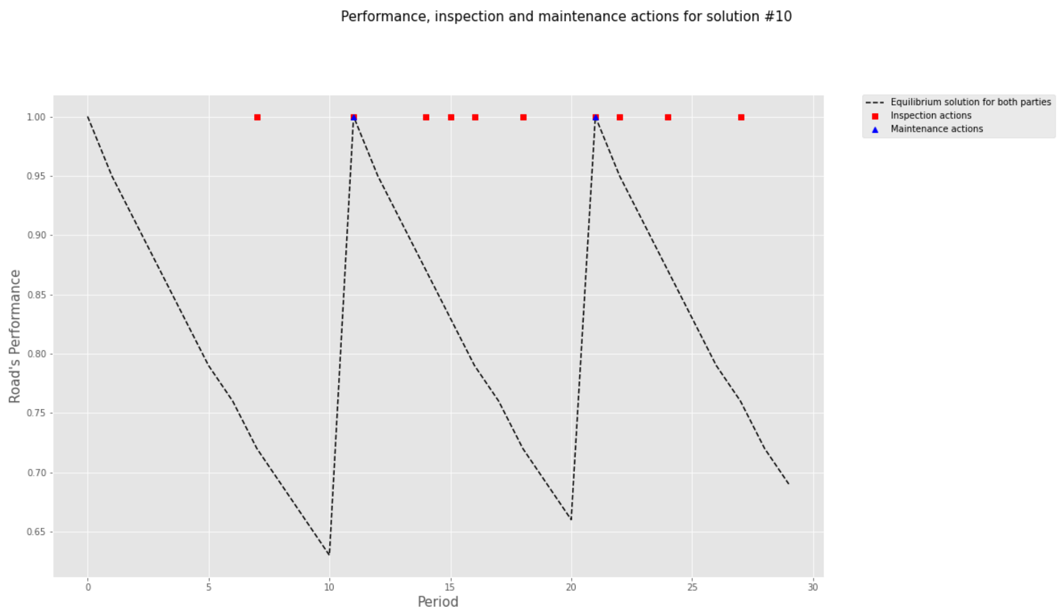

Fig. 36. Solution 10

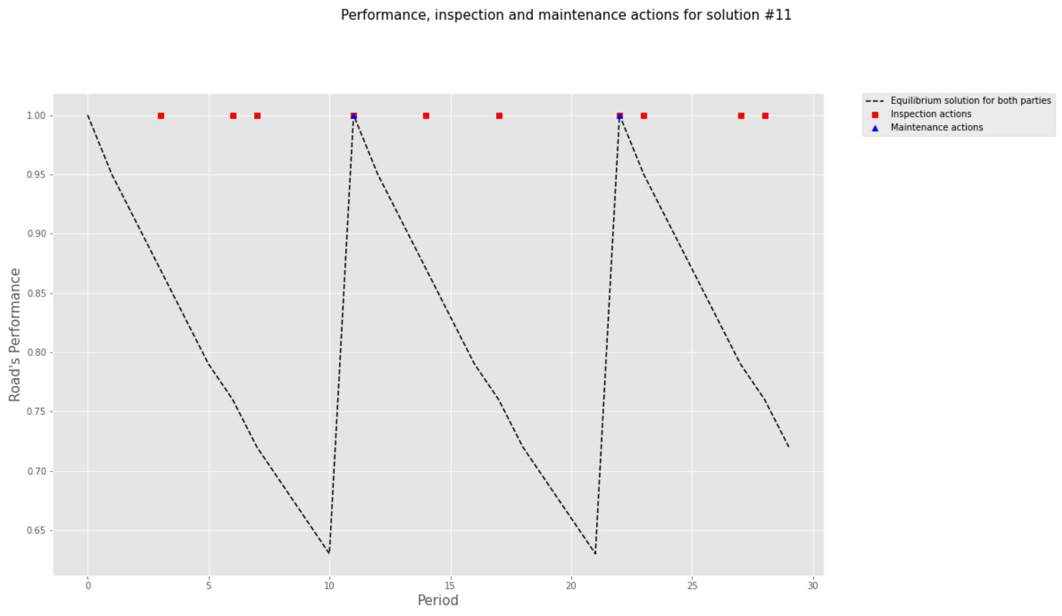

Fig. 37. Solution 11

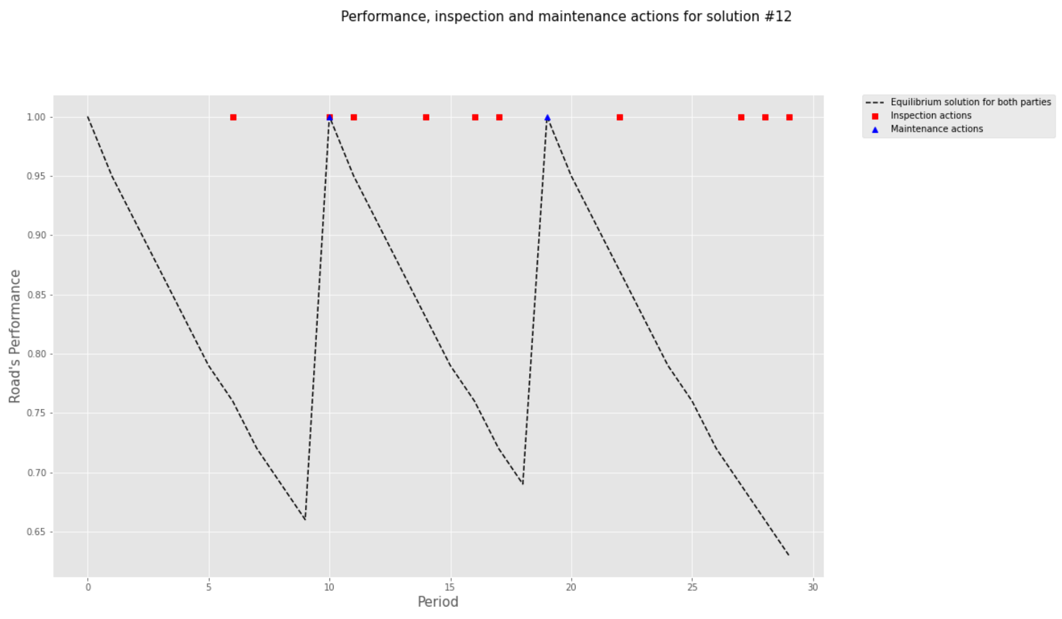

Fig. 38.   Solution 12

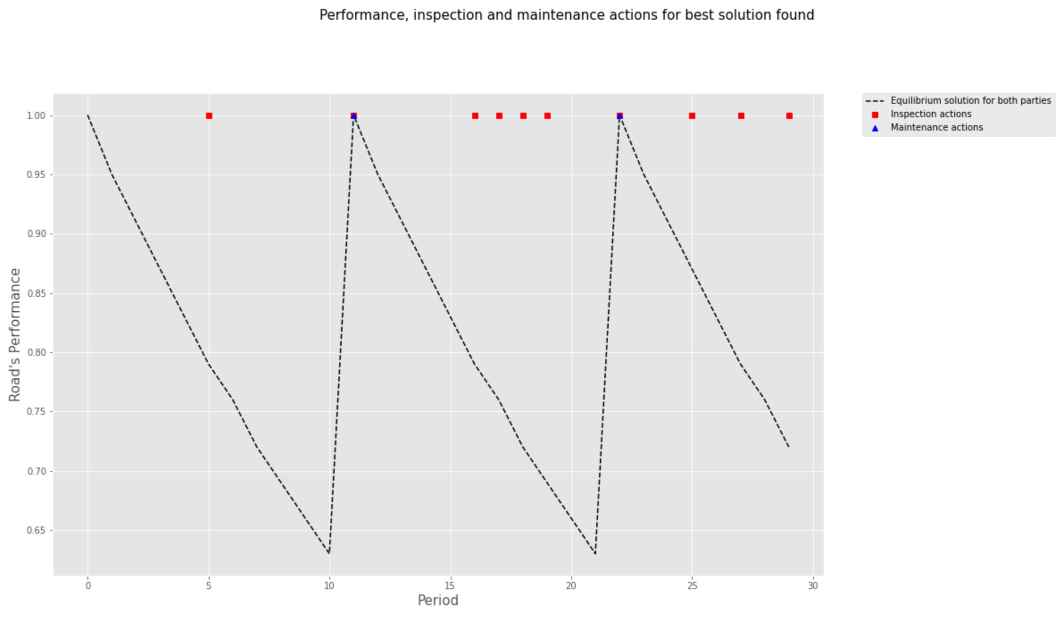

Fig. 39.   Solution 13