TOML Project 2

Oriol Martínez Acón

May 2022

1 Introduction

In this problem we have the following statement. We have to deploy a sensor network connected wireless. Each of the sensors captures different physical phenomenons and they are sent to a gateway connected to the Internet. The packet will be re-transmitting through the different sensors till it reaches the gateway. The **Media Access Protocol** (MAC) specifies when there will be transmission of the packets. The MAC protocol follows a duty-cycle type architecture, which consists of being in a sleep state for an amount of time, until it is the turn to transmit a packet. The **sleep state** consumes less energy what it translates into better autonomy. There are three different transmission states: *carrier sense*, *receiving* and *transmission*, all these states consumes at least ten times more energy than the **sleep state**. The main goal for the project is how to fix the amount of time to be sleep/awake in order to make the battery live longer or just to minimize the delay in the transmission of the packets. The network architecture has a random topology. All the nodes remain static in the system and located in a circle distribution. The transmission of the packets follows the shortest path to arrive to the gateway. So, every source node generates packets with frequency F_s. This parameter can be found by doing $F_s = 1.0/(min * 60 * 1000)$ where the min are the number of minutes. The parameter d describes how far is the node to arrive to the destination. A $d = D$ means the node is located in the furthest ring from the gateway (center node), while, a $d = 0$ means the node is at the center node. The following image depicts perfectly the concepts explained.

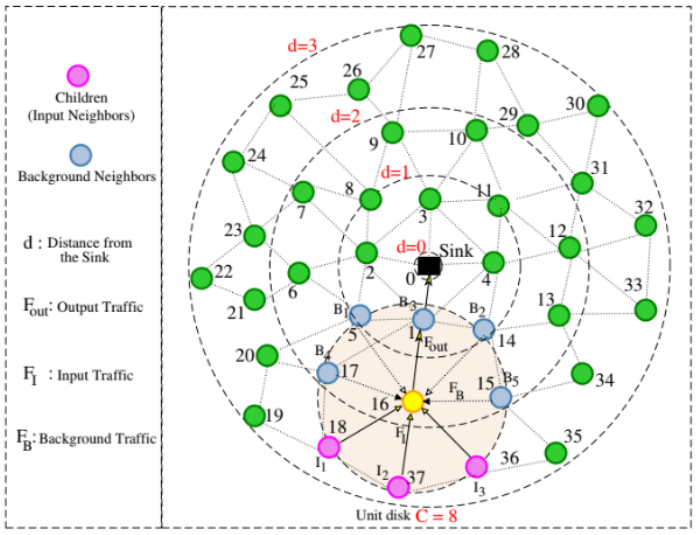

Figure 1: Typical Network Topology with Local Node Traffic Flowing.

The X-MAC protocol is a protocol that sets a time variable where the node will wake up to transmit the packets, this variable is T_w. So as it was said before the idea is to quantify the impact of the variable in the system in terms of delay and energy consumption. An emergent system with a high delay could be translated into a catastrophic event, or in the other hand, a system where the importance of the transmission of the packet is not important would be better have a good management of the energy consumption. In the image below we can have a good overview about X-MAC protocol.

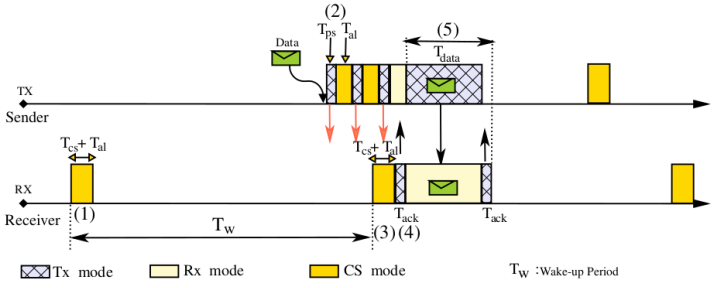

Figure 2: X-MAC's carrier sensing, transmission, and receiving modes.

All the code developed in the project can be found in: `https://github.com/oriolmartinezac/TOML-Labs/tree/main/project-2`

2 Energy and delay functions

2.1 Energy function

The energy function depicts how the node (sensors) are consuming the charge of the battery. The energy function depends by the values of α_1, α_2, α_3. The expression of the energy equation can be defined as:

$$E^{XMAC} = \frac{\alpha_1}{T_w} + \alpha_2 T_w + \alpha_3$$

Each of these values also depends of the packets that are transmitting. To calculate all these parameters we need to have into account:

- Node's output traffic frequency at level d, F^d_{out}.
- Node's input traffic frequency at level d, F^d_I.
- Background node's traffic frequency at level d, F^d_B.

All the parameters depend of the F_s, the sampling rate [pkt/node/min], and d the distance level of the rings in reference to the central node. The F_s is calculated by a list with the number of minutes [1, 5, 10, 15, 20, 25, 30]. The first thing to do in order to understand the problem is to represent the functions in a graphic fashion. In the following image we can see the results if we modify the F_s with energy function (*ylabel*) compare to T_w (*xlabel*) that will take the values from 100ms to 500ms.

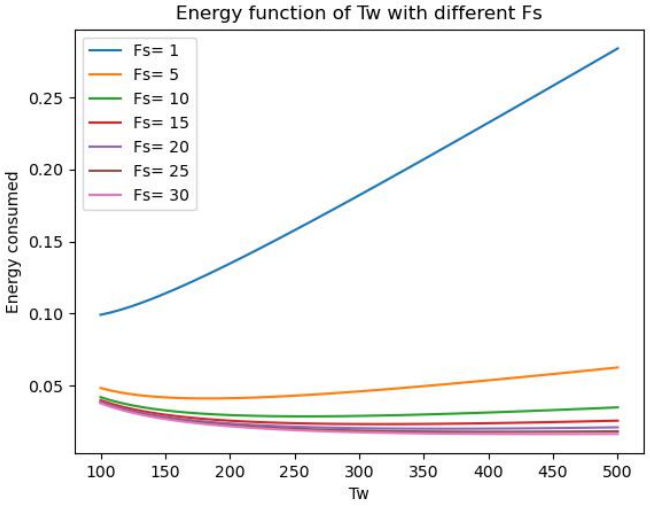

Figure 3: Plots of the energy function from $F_s = 1$ to $F_s = 30$.

2.2 Delay function

The delay function depicts how the message takes to move between different nodes. The delay function depends by the values of β_1, β_2. The function is defined as:

$$L^{XMAC} = \beta_1 * T_w + \beta_2$$

To calculate the *beta* parameters we need to have into account the d, Tcw and $Tdata$ values. Like with the energy function, in order to understand the problem is better to plot, in this case the delay function is shown below. The values for the T_w, x-axis chosen are from 100ms to 500ms.

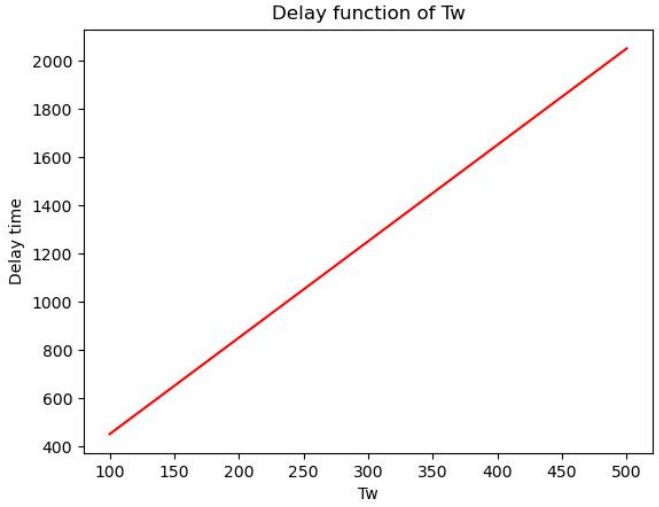

Figure 4: Plot of the delay function.

In the plot of the function we can see that the growth of the the delay time by the T_w values follows a straight line relation.

2.3 Energy function vs delay function

Finally, to get a good overview of the problem is better to plot the energy function vs the delay function. In this plot we can see how the F_s values can affect to the delay and energy consumption in the problem. The energy function and the delay function are inversely related that means that for a value of T_w that gives us a good result in energy function will not give the same good value in terms of delay and, actually, a really bad result.

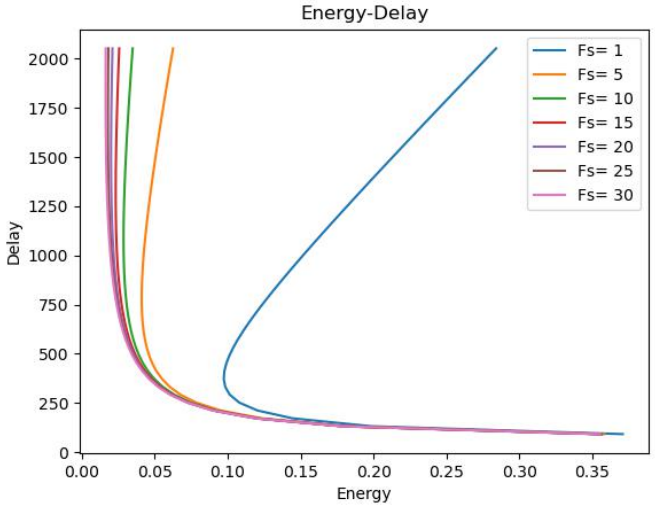

Figure 5: Plot of the energy function against delay function.

3 Optimization problems

3.1 Energy optimization

In this section the idea is to find the minimum of energy function (optimize) using a solver. The problem to optimize follows the next structure.

$$\begin{aligned} \text{minimize} \quad & E^{XMAC}(T_w) \\ \text{subject to} \quad & L^{XMAC}(T_w) \leq L_{max} \\ & T_w \geq Tmin_w \\ & |I^0|E^1_{tx} \leq 1/4 \\ \text{var} \quad & T_w \end{aligned}$$

Even it seems an easy problem to optimize, there is an adversity, the problem to optimize is **Non-convex** but **Geometric Programming** (GP), is for this reason that scipy and cvxpy solvers (used in previous lab) can not solve this problem. The solver used has been the *gpkit*, is important to remark that the solver is not totally free to use. The following images depict the results of minimizing the energy problem with different L_{max} values, [100, 500, 1000, 2000, 3500, 5000].

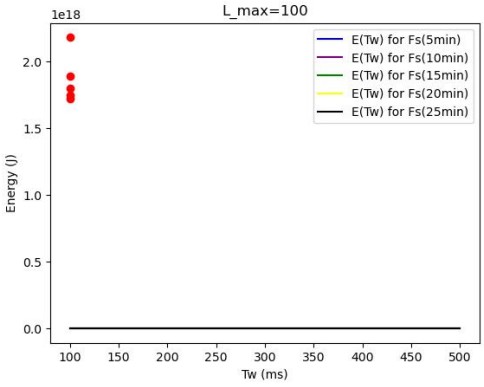

Figure 6: Energy optimization with $L_{max} = 100$.

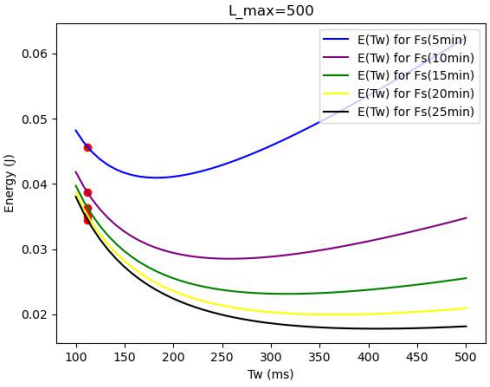

Figure 7: Energy optimization with $L_{max} = 500$.

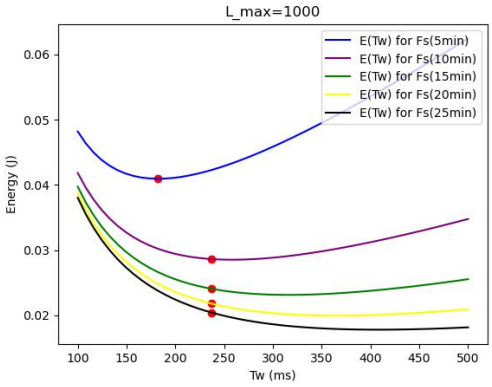

Figure 8: Energy optimization with $L_{max} = 1000$.

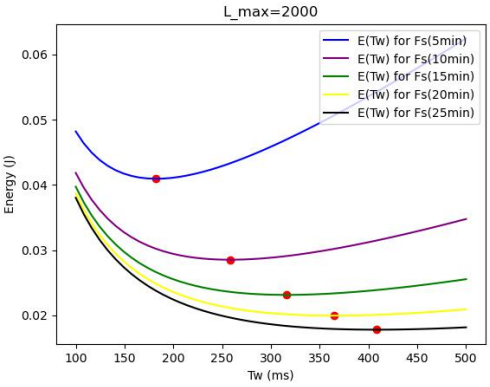

Figure 9: Energy optimization with $L_{max} = 2000$.

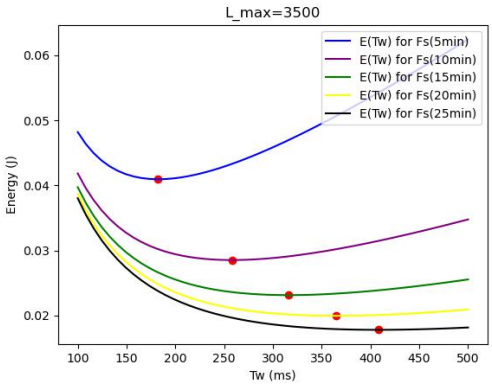

Figure 10: Energy optimization with $L_{max} = 3500$.

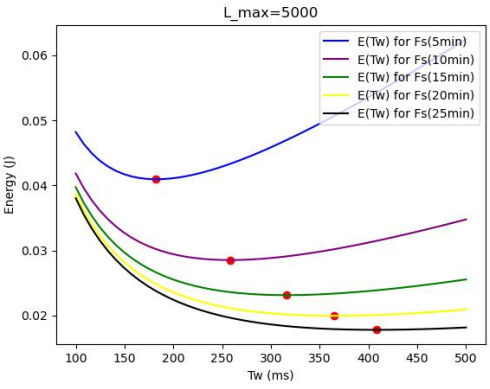

Figure 11: Energy optimization with $L_{max} = 5000$.

As we can see in the energy optimization result with $L_{max} = 100$ the problem has no feasible solution. Is only when $L_{max} \geq 500$. Another thing that we can see in the plots is that the red dot, solution cost, gives the minimum point in the energy function. From $L_{max} = 2000$ all the energy function has the same minimums, so for the NBS it will be taken any value of the $L_{max} \geq 2000$. In addition, the lowest point for those graphs is obtained with $F_s = 25$ pkts/min. The minimum of the energy function with that values is $E = 0.017812055765130203$.

3.2 Delay optimization

In this optimization problem there is exactly the same problem as in the energy optimization problem. The problem to optimize is **Non-convex** but **Geometric Programming** (GP), is for this reason that scipy and cvxpy solvers (used in previous lab) can not solve this problem. The solver used has been the *gpkit*. In the following image there are all the results of executing the optimization problem with different E_{budget} values, [0.5, 1.0, 2.0, 3.0, 4.0, 5.0].

```
81
82 E_budget = 0.5
83 Tw = 100.00000105436011
84 p*= 452.0479527477433
85
86 E_budget = 1.0
87 Tw = 100.00000380620071
88 p*= 452.04783780150484
89
90 E_budget = 2.0
91 Tw = 100.00000003871183
92 p*= 452.04799465102684
93
94 E_budget = 3.0
95 Tw = 100.00000004761058
96 p*= 452.0479899434953
97
98 E_budget = 4.0
99 Tw = 100.0000000541173
100 p*= 452.04798516325263
101
102 E_budget = 5.0
103 Tw = 100.00000005923826
104 p*= 452.047980505379
105
```

Figure 12: Delay optimization results with all the different $E_{budgets}$

As we can see the best result is when $E_{budget} = 1.0$, because it has the lower solution cost. Once, the results are obtained a good way to understand what they mean is to plot them.

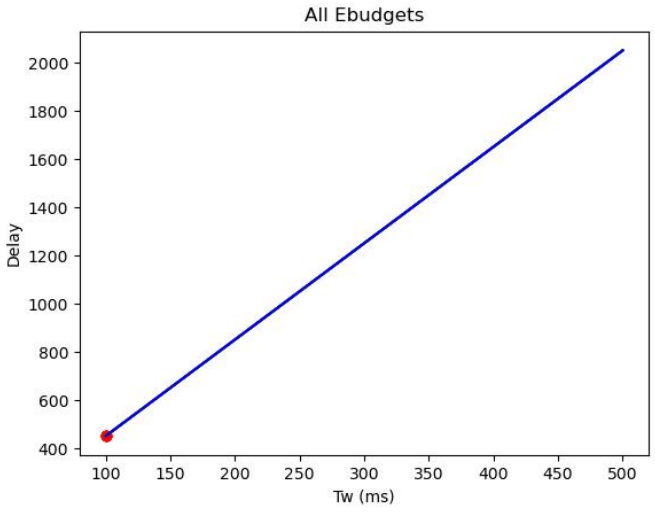

Figure 13: Delay optimization with all E_{budget} values.

4 Game theory

In the initial problem the energy function and the delay function are inversely related, i.e. if the value that minimizes the energy function maximize the delay function. In this section the idea is to implement **Nash Bargaining Solution** (NBS) in order to find a trade-off between the energy function and the delay function. The NBS starts from a simple premise both players that wants to find a converging point must to be rational, otherwise there is totally impossible to find a solution because both functions want to take their best value and then a point not feasible. The NBS is not the only way to find a trade-off between two different players, in terms of game theory, there is another called **Kalai-Smorodinsky Bargaining Solution** (KBS). The difference between them is that the NBS set axioms to find a converging point in equilibrium with both players; on the other hand, the KBS set axioms that empower the player who has more value to threat, in other words, is giving more importance to the slope of the function. E.g. in the field of war, a country with high weapon power has more to threat than another with less and the convergence point will be more favorable for the one with more power. In the project the NBS is the one chosen to find the trade-off, the problem to optimize is the following one. The

formalization of the problem is the following

$$
\begin{aligned}
\text{maximize} \quad & log(E^{XMAC}_{worst} - E_1) + log(L^{XMAC}_{worst} - L_1) \\
\text{subject to} \quad & E^{XMAC}_{worst} \geq E^{XMAC}(T_w) \\
& E_1 \geq E^{XMAC}(T_w) \\
& L^{XMAC}_{worst} \leq L^{XMAC}(T_w) \\
& L_1 \geq L^{XMAC}(T_w) \\
& T_w \geq T^{min}_w \\
\text{var} \quad & E_1, L_1, T_w
\end{aligned}
$$

As we can see the problem is concave, in order to simplify the problem for further computation, the problem has been changed to a convex one. Here is the formalization problem changed.

$$
\begin{aligned}
\text{minimize} \quad & - log(E^{XMAC}_{worst} - E_1) - log(L^{XMAC}_{worst} - L_1) \\
\text{subject to} \quad & E^{XMAC}_{worst} \geq E^{XMAC}(T_w) \\
& E_1 \geq E^{XMAC}(T_w) \\
& L^{XMAC}_{worst} \leq L^{XMAC}(T_w) \\
& L_1 \geq L^{XMAC}(T_w) \\
& T_w \geq T^{min}_w \\
\text{var} \quad & E_1, L_1, T_w
\end{aligned}
$$

The **worst** and **best** values used to solve the problem are obtained from the previous section. $E_{worst} = 0.04568$, $E_{best} = 0.01781$, $L_{best} = 452.04800$ and the L_{worst} will be the dynamic parameter too see the impact in the NBS problem. Is important also to highlight that the F_s taken is from the best solution from the previous section, $F_s = 25min$. The results obtained has been plot in order to make a good analysis. The executions computed has been with different L_{max} values, [500, 750, 1500, 2000, 3500]. Below there are the results obtained after executing the solver with the statement defined previously.

```
 8 LMAX=500
 9 optimal value p* =  1.2543821690045829
10 optimal var: E_1 =  0.03973534704992206
11 optimal var: L_1 =  452.0480284816025
12 optimal var: T_w =  100.00000512880833
13
14 LMAX=750
15 optimal value p* =  -0.8885147275287597
16 optimal var: E_1 =  0.03441732038421913
17 optimal var: L_1 =  534.1862189153597
18 optimal var: T_w =  120.5345110126341
19
20 LMAX=1500
21 optimal value p* =  -2.631219800203845
22 optimal var: E_1 =  0.027859904248847194
23 optimal var: L_1 =  720.6805476472834
24 optimal var: T_w =  167.15800865066186
25
26 LMAX=2000
27 optimal value p* =  -3.1461529794477077
28 optimal var: E_1 =  0.02625162398898684
29 optimal var: L_1 =  803.7282228423993
30 optimal var: T_w =  187.9197409222023
31
32 LMAX=2500
33 optimal value p* =  -3.5028527512116447
34 optimal var: E_1 =  0.025339934876423626
35 optimal var: L_1 =  867.5840956400455
36 optimal var: T_w =  203.8838479012106
37
38 LMAX=3500
39 optimal value p* =  -3.9906383902913025
40 optimal var: E_1 =  0.0243987078748298
41 optimal var: L_1 =  958.842411696147
42 optimal var: T_w =  226.69819888838913
43
```

Figure 14: Outputs from the solver applying the NBS to the problem.

In the images below there are respective plots of the results. To have a good presentation the best values and worst used in the execution has been shown in the plot too.

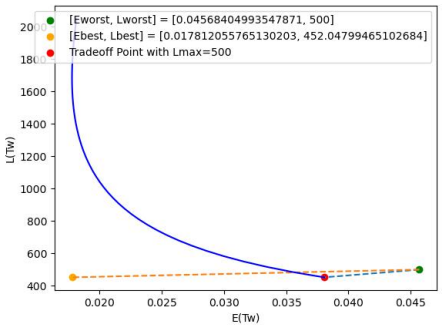

Figure 15: NBS with $L_{max} = 500$.

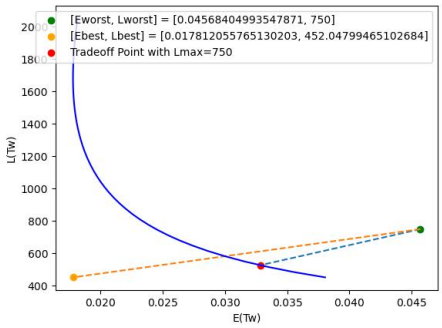

Figure 16: NBS with $L_{max} = 750$.

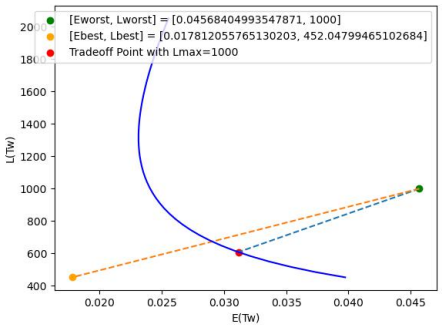

Figure 17: NBS with $L_{max} = 1000$.

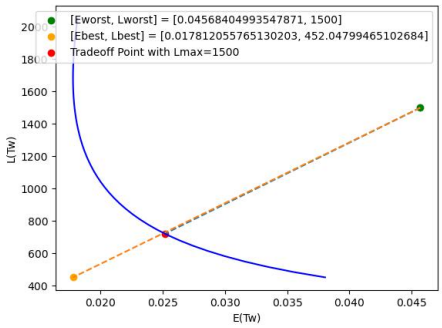

Figure 18: NBS with $L_{max} = 1500$.

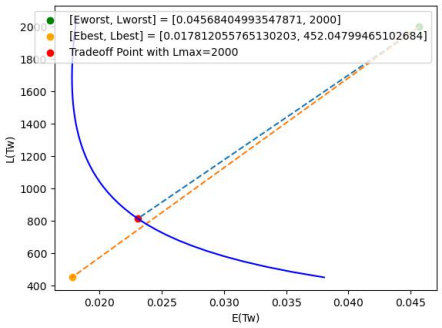

Figure 19: NBS with $L_{max} = 2000$.

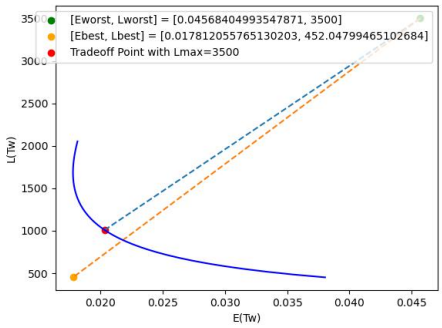

Figure 20: NBS with $L_{max} = 3500$.

As we can see in the figures the NBS is finding an trade-off point in equilibrium with both players. Finally in the following plot we can see more different trade-offs together in the same graph, $L_{max} = [500, 1000, 2000, 3000, 4000, 5000]$.

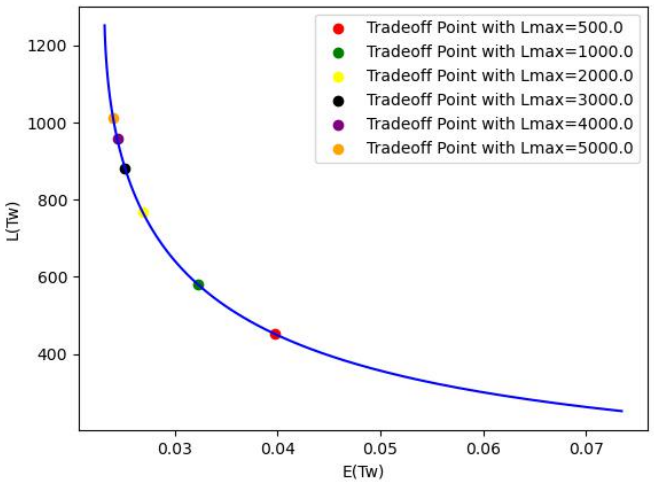

Figure 21: Outputs from the solver applying the NBS to the problem.

This last plot is useful to see how the solution is getting worse in terms of balance by changing the L_{max} value. So, in conclusion the best value that you can take is when $L_{max} = 2000$ because is more close to have an equilibrium between the delay function and the energy function.

References

[1] Theory from the slides of subject Topics on Optimization and Machine Learning. José M. Barceló Ordinas, Departament d'Arquitectura de Computadors (UPC).

[2] Wikipedia contributors. Cooperative bargaining [Internet]. Wikipedia, The Free Encyclopedia. 2021. Available in: `https://en.wikipedia.org/w/index.php?title=Cooperative_bargaining&oldid=1056609571`

[3] Doudou M, Barcelo-Ordinas JM, Djenouri D, Garcia-Vidal J, Bouabdallah A, Badache N. Game theory framework for MAC parameter optimization in energy-delay constrained sensor networks. ACM Trans Sens Netw [Internet]. 2016;12(2):1–35. Available in: `http://dx.doi.org/10.1145/2883615`

[4] Readthedocs.io. [citado el 8 de mayo de 2022]. Available in: `https://gpkit.readthedocs.io/_/downloads/en/latest/pdf/`