FastRAO

1

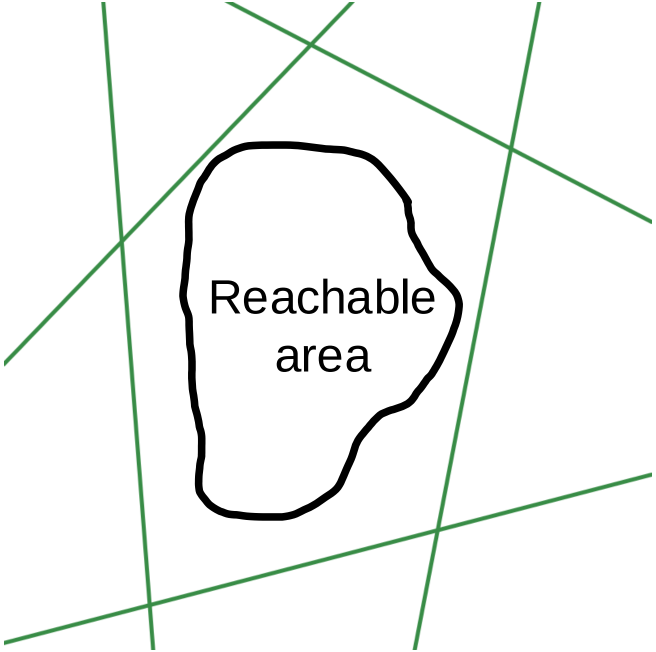
Observation

Observation

In a general case, the level of congestion on the different lines and for different states is very varied.
This means that some CNECs will have a positive margin for any combination of remedial actions.

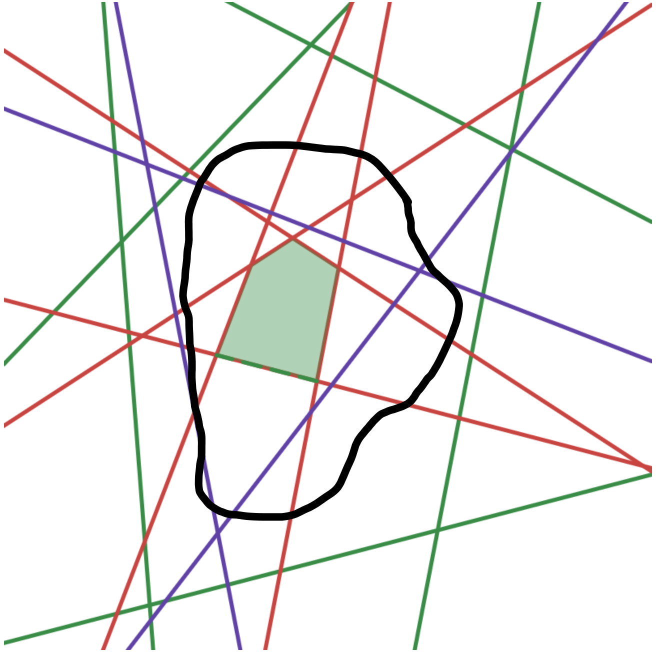
Idea: There is no point considering those CNECs in the RAO because they will always be secure no matter the combination of actions we choose.

Observation



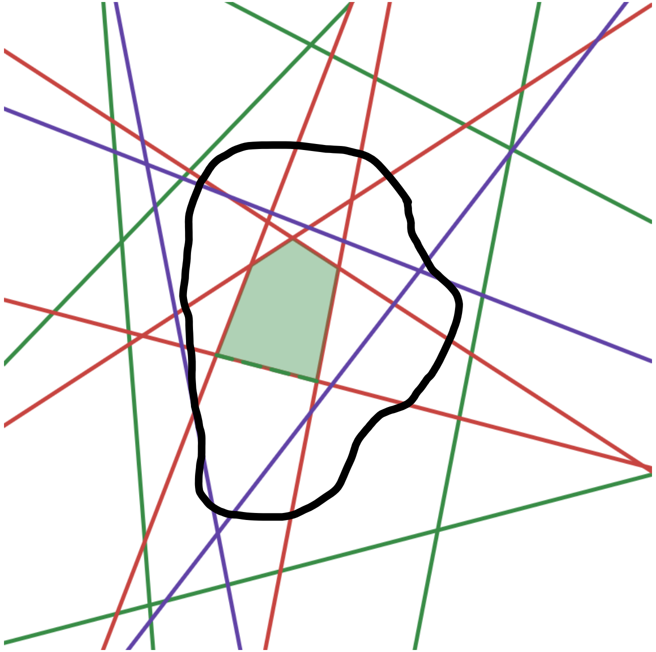
Here we represent a set of CNECs which are always secure (the limit of each CNEC is represented by a line).
These CNECs can be ignored by the RAO.

Observation



This is a more complex case where we show all the CNECs.
The green area is the secure area, where all the CNECs are secure. The goal of the RAO is to find the best combination of actions (which will obviously be in that area).

Observation



We can actually distinguish three kind of CNECs:

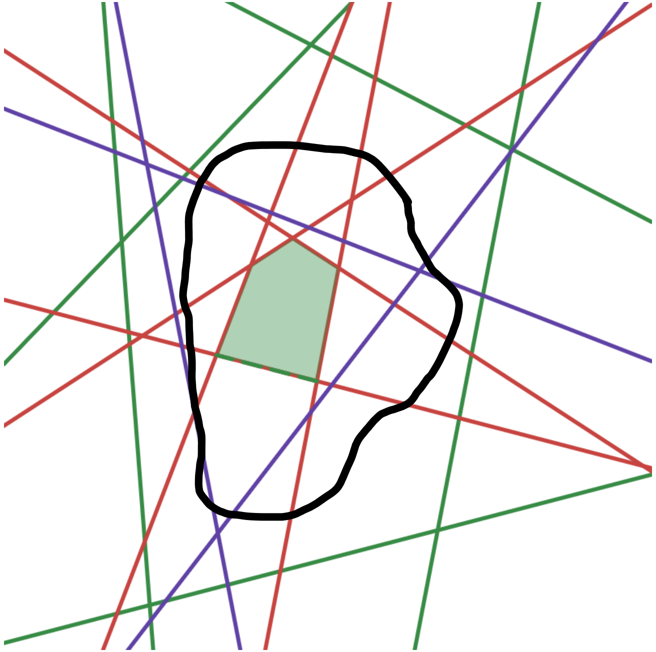
- CNECs which are always secure (green)
- CNECs which define the boundary of the secure area (red)
- Other CNECs (purple)

Only the red ones are interesting for the RAO.
(If we have a situation where a purple CNEC is unsecure, then at least one red CNEC is also unsecure)
In an ideal world, we would only run the RAO on these red CNECs.

2

Algorithm

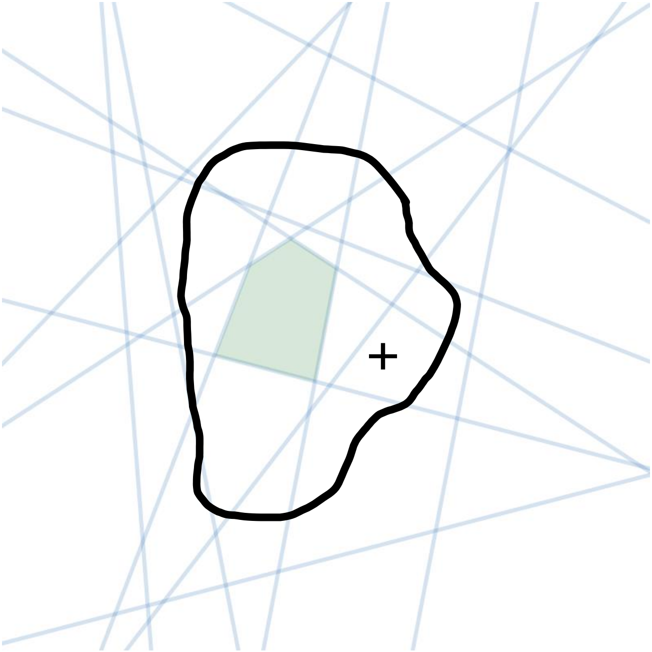
Algorithm



The goal is to run the RAO on only critical CNECs to accelerate the computation.

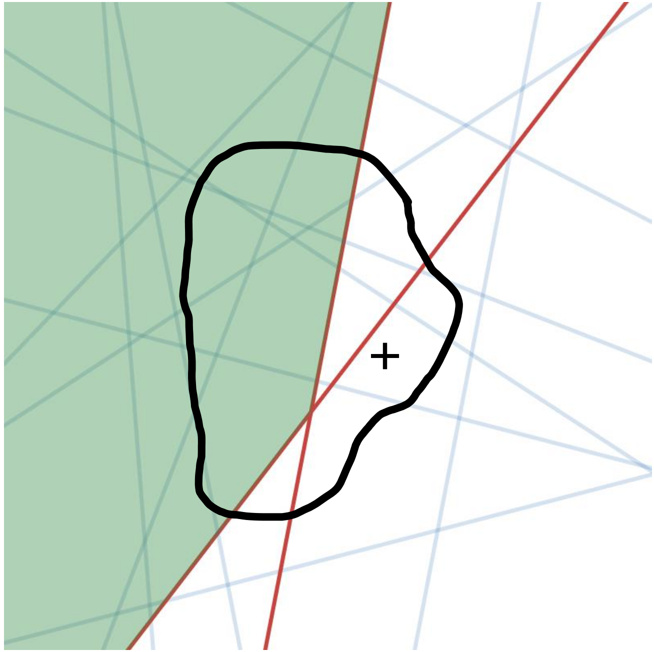
But we do not know in advance which these critical CNECs are.

Algorithm



We start by running a loadflow to compute the margin on ALL CNECs in the initial situation (+)

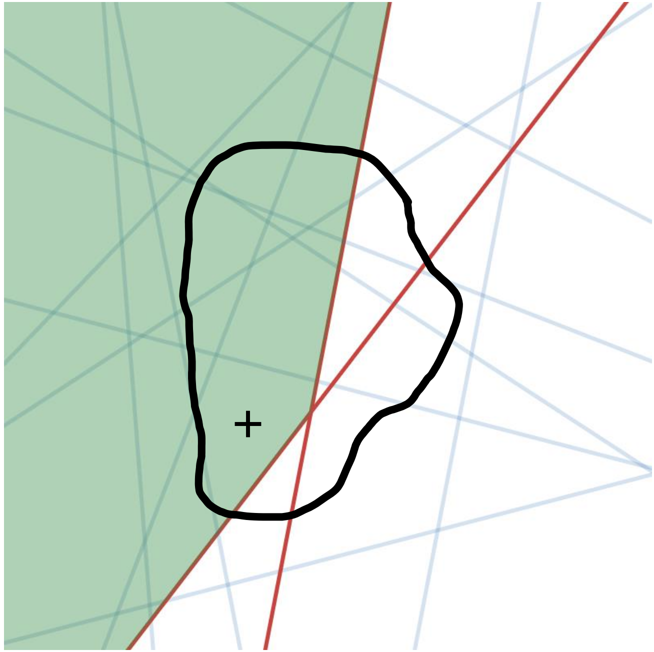
Algorithm



In the initial situation (+) we have two CNECs which are unsecure.

We will run a RAO on these two CNECs.
The solution of this RAO will fall in the green region.

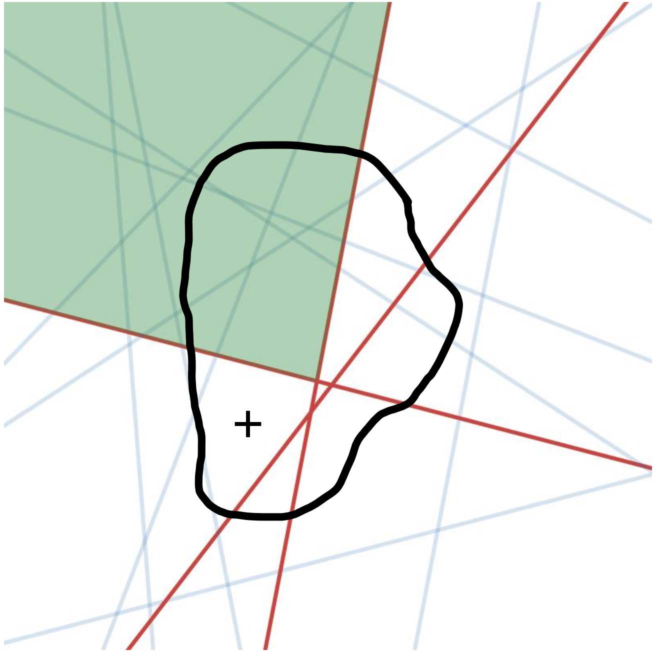
Algorithm



The RAO found a solution (+).

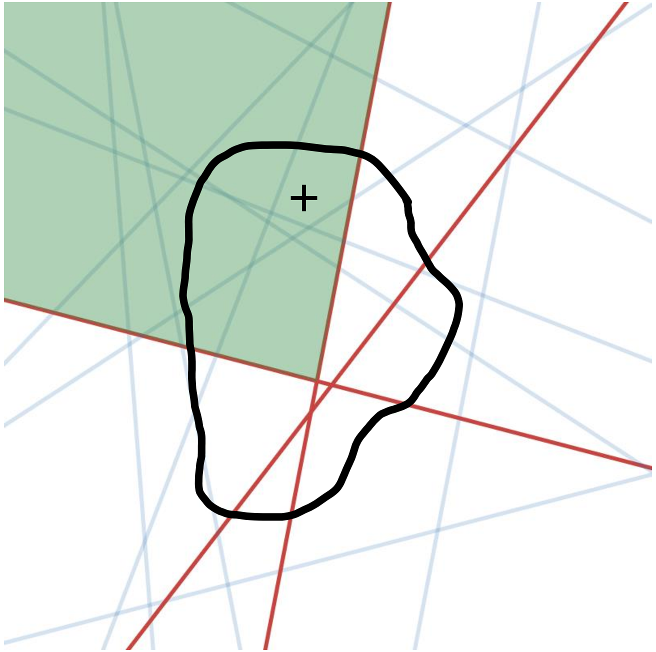
We apply that solution to the network and we run a new loadflow on ALL CNECs to check if any of them are unsecure.

Algorithm



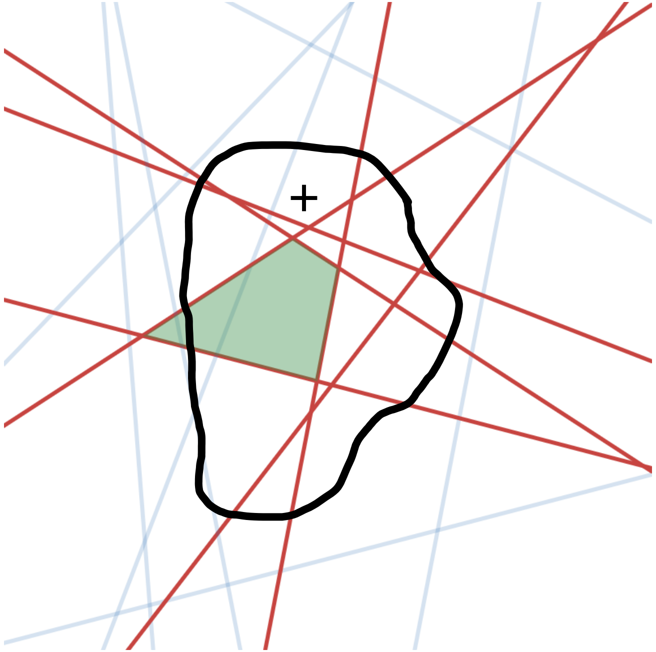
We have once again an unsecure CNEC. We will therefore have to run a new RAO, including this CNEC (and the previous ones).
The result will fall in the new green area.

Algorithm



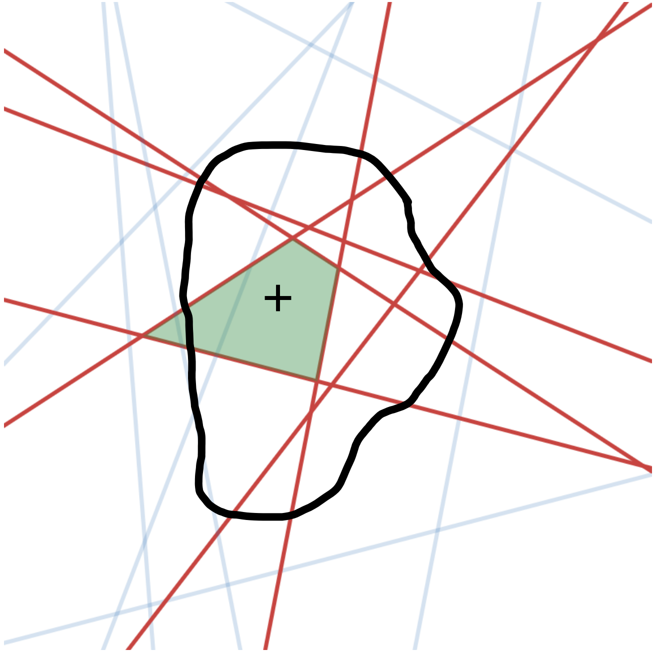
The RAO finds a new solution (+) and we run a new loadflow with the new remedial actions applied.

Algorithm



We have a few unsecure CNECs which we add to the set of critical CNECs.
Once again we have to run a RAO.

Algorithm



The RAO found a new result (+) and a new loadflow on ALL CNECs is required.

This time all CNECs are secure. We can stop.

Algorithm

Remarks:

- If no secure solution existed, there would be a RAO which does not succeed in securing the set of critical CNECs for that given step, and we would end there knowing that the network can not be secured (not implemented yet)

- We can adapt the algorithm to maximize the worst margin instead of securing the network.

- In reality there are orders of magnitude (approximately 100x to 1000x) between the number of "critical cnecs" and the "non-critical" cnecs.

- If we need to do multiple RAOs on the same set of CNECs and very similar networks (for instance a dichotomy for capacity calculation), we can re-use the set of critical cnecs to initialize the next RAO.

3

POC

POC

Remarks:

- The POC was tested on our non regression test data and gave very similar results (most cases gave exactly the same result (same actions and exactly the same flows), max difference was only a few MW).

- For the following performance tables, SWE CC data was used. In the tested individual cases, exactly the same results were found. For the dichotomy runs, exactly the same steps were secure / unsecure.

POC

	Classic RAO	Fast RAO
Case 1	5:25	2:58
Case 2	11:12	4:26
Case 3	7:23	3:46
Case 4	9:33	8:03
Case 5	1:22	2:28
Case 6	4:04	6:04
Case 7	6:52	5:50

Running a single RAO without
initializing the critical cnecs

POC

	Classic RAO	Fast RAO
Case 1	37:13	35:53
Case 2	50:07	32:54
Case 3	32:46	24:28
Case 4	40:04	16:40

Running a full dichotomy and using previous step critical cnecs for initializing

POC

TS	Computation time Castor	Is secure Castor	Computation time Fast RAO	Is secure Fast RAO
20231226_1930 2778 MW FR->ES	59s	false	1min 39s	false
20231226_1930 2778 MW FR->ES no curative if preventive unsecure	1min	false	1min 39s	false
20231226_1930 2828 MW FR->ES	1min 45s	false	2min 59s	false
20231226_1930 2828 MW FR->ES no curative if preventive unsecure	35s	false	22s	false
20231226_1930 2928 MW FR->ES	2min 46s	false	3min 17s	false
20231226_1930 2928 MW FR->ES no curative if preventive unsecure	40s	false	22s	false
20231226_1930 3128 MW FR->ES	1min 49s	false	3min 44s	false
20231226_1930 3128 MW FR->ES no curative if preventive unsecure	32s	false	25s	false
20240215_0930 2650 MW ES->FR	15min 45s	false	1min 31s	false
20240215_0930 2650 MW ES->FR no curative if preventive unsecure	1min 1s	false	15s	false
20240106_0530 3200 MW ES->PT	25s	true	15s	true
20240106_0530 3250 MW ES->PT	25s	true	15s	true
20240106_0530 3300 MW ES->PT	26s	true	15s	true
20240106_0530 3500 MW ES->PT	25s	true	15s	true

Results for BTCC data (individual runs)