Methodology to obtain “near-optimal solutions” (How the modified script works)

The script starts with finding the optimal solution. It accepts the normal sets & parameters excel files as an input, and outputs the usual results. From this run (Run_0), some results are particularly important: The total cost of the run, and the most dominant technology throughout the time domain of the optimization. The total cost is the sum of all the costs of the run, stored in the result excel file. To identify the most dominant technology, the results for new capacity added is used. The script analyses all the power generation blocks one by one and calculates the sum of new capacity added for that technology throughout the years. Then, it compares this summation between the technologies and identifies the most dominant one. The script also remembers the sum of the capacity added for that technology, and also the total capacity of that technology at the end of the time domain. After that, the script selects a random technology among the ones remaining, and also saves the sum of the new capacity and the total capacity at the last time period of that random technology.

After that, the parameter modification starts. A new variable called “Crossover variable” (C) is introduced. This variable is the base of the methodology. This variable uses the saved values of the new capacities of the most dominant and the random technology. It is basically equal to the average of the two values. The Crossover value is used to apply a further constraint to the run, making the sequential runs “not optimal”. We apply a value to limit the maximum total capacity of the most dominant technology of the previous run. We have already saved the total capacity of that technology, and we implement a maximum limit equal to the total capacity of the technology minus the crossover variable. This way, we are forcing the most dominant technology to be used less than the optimal run. It is understood that with this method only, we will reach repeating optimal solutions, as we work only with the most

dominant technology, which is going to be the same thing in most runs. To solve this problem, we add a property of randomness to the method. This randomness will lead to obtaining new solutions even after hundreds of run, while also helping us navigate through the solution space more effectively.

Hence, we apply a minimum capacity limit to a random technology selected on the previous run. This minimum capacity value is equal to the total capacity of this random technology, saved on the previous run, plus the crossover variable. This way, we are forcing a random technology to be implemented with a certain value, helping us add diversity to our near-optimal solutions and explore the corners of the solution space more effectively.

The script modifies the parameters excel file on its own and runs the optimization. If this run is infeasible and has no results, the script modifies the crossover variable by reducing the value of the variable by 10%. This way, the constraint becomes less strict, and we get closer to the initial optimal solution, so closer to have a feasible result. If the run has some results, we check the total cost. The user sets a slack in percentage before initialization of the code, that basically sets a limit on the maximum total cost acceptable. If the total cost of the run is higher than the optimal total cost plus the slack, the run is unacceptable, and the crossover variable is again modified in the same way, resulting in a run closer to the optimal run, and therefore, with lower total cost. Finally, when we have a feasible run within the slack range, the run is counted successful, and we have a near-optimal solution. The whole process will go again, using the results from the previous run: A new most dominant technology, a new random technology, new values and new crossover variable, resulting in a new near optimal solution each time. The number of solutions can technically be infinite, so the user will have to define the number of solutions needed.