Application of multi-processing programming for transmission planning at scenario reduction stage using python

1 Introduction

To study planning for transmission system expansion, there is a need to include a large number of scenarios. However; it is not possible to consider all of them in the optimization process as it makes optimization run unacceptably long, and, the problem intractable. To reduce the number of under-studied scenarios, a scenario reduction stage is required. In this stage, those scenarios which are similar to each other are classified in different groups and for each group, one representative scenario will be found. Meanwhile, this process is often time-consuming as there are a large number of scenarios among which representative scenarios should be selected.

This document is a report on the application of multi-processing programming in acceleration of the above mentioned procedure using fast forward selection algorithm.

2 Problem description

In this work, fast forward selection (FFS) algorithm is utilised for scenario reduction. This algorithm selects a pre-determined number of scenarios which can be the best representatives of the others. A weight is assigned to each representative scenario implying the probability of the occurrence of that particular scenario. Similarity is measured with distance between different scenarios which, here is based on calculation of Euclidean norm for different vectors created for the mathematical evaluation of the scenarios. For more information on FFS algorithm please refer to [1] .

This study is a planning for ten years including dynamic line rating systems (DLR). Three different regions are defined in the network and for these regions, historical data for 2010-2019 relating to weather stations in Ireland is used, including wind speed and temperature. Maximum value for solar irradiation is also calculated based on the relations existing in [2] .

Each year contains 8760 hours and totally, 200 representative hours (scenarios) are needed to be selected per year. Each scenario involves 3X2 + 1 + 1 = 8 parameters (wind speed and temperature for three regions, one solar irradiation and total network load). Therefore, each scenario vector has 8 dimensions.

3 Distance matrix calculation

Euclidean norm is considered in this work for calculation of the distance between different scenarios. For evaluation of distance matrix for scenario vectors two different approaches are utilised and examined comparatively:

1- Iteration through scenario vectors and using ‘numpy.linalg.norm’ function for each scenario pair
2- ‘SciPy.spatial.distance_matrix’ function

4 Results

The calculation is done using the server PC with characteristics of Intel® Xeon® Processor E5-2697 v2 12 cores, 256GB RAM. Four distinct approaches are tested:

1- Single-processing implementation and evaluation of distance matrix using iteration
2- Multi-processing implementation for the evaluation of distance matrix using iteration
3- Single-processing implementation and evaluation of distance matrix using SciPy package
4- Multi-processing implementation for individual years and evaluation of distance matrix using SciPy package

Ten processes are used for multi-processing implementations. The link between these processes and the main process is based on client/server communication (socket programming) which makes initialization stage slightly long.

The results for the implementation of the above approaches are depicted in Fig.1-4. To track time length for each stage, a 'timing' class is coded to ease the representation of real time execution of different stages, which its output is also brought in the figures. At last, run time results are summarized in Table.I.

Table I. Summary

	Parallelism	Distance matrix calculation	Overall run time (s)
	Multi/Single	Iteration/SciPy	
Approach I	Single	Iteration	11067
Approach II	Multi	Iteration	2220
Approach III	Single	SciPy	1385
Approach IV	Multi	SciPy	280

5 Conclusion

Multi-processing programming can considerably accelerate the process of the scenario reduction, specifically, when it accompanies with prudent utilisation of high-performance numpy and sciPy functions for matrix operations, leading to a reduction of run time for this study case from 3 hours to less than 5 minutes. This study is typical and in real cases, problem dimensions are much larger and therefore, time saving using this approach will be more discernible.

Generally, multi-processing programming is suggested when bottleneck of the implementation inevitably consists of python codes. Meanwhile, it is not efficient for the cases in which the whole algorithm can be implemented using high-performance numpy and sciPy functions. In these cases, running the code inside single process is faster and does not need involving the complexity of multi-processing programming.

The combinational solution resulted in the lowest run time as the fast forward algorithm for the scenario reduction cannot be implemented using numpy and sciPy functions alone and requires iterations using python codes.

6 References

[1]. A.J. Conejo, M. Carrion, J.M. Morales. "Decision making under uncertainty in electricity markets", Springer US, 2010

[2]. "IEEE standard for calculating the current-temperature relationship of bare overhead conductors", IEEE Std 738-2012, Dec. 2013

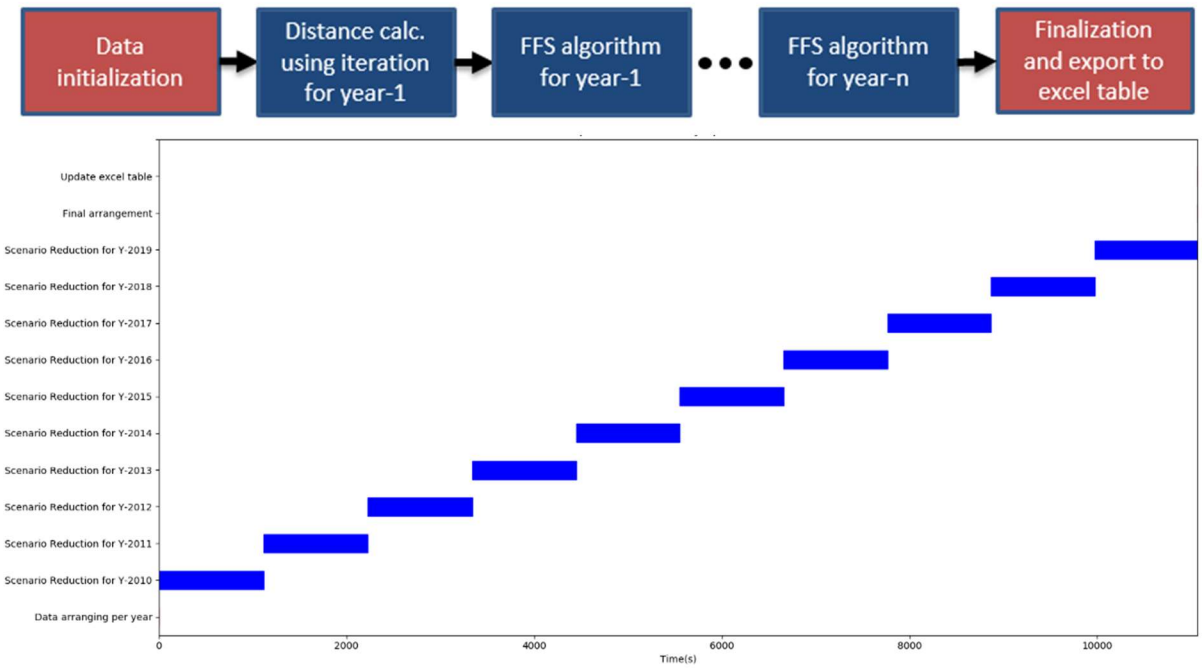

Fig.1 Single processing implementation and iteration-based distance matrix calculation

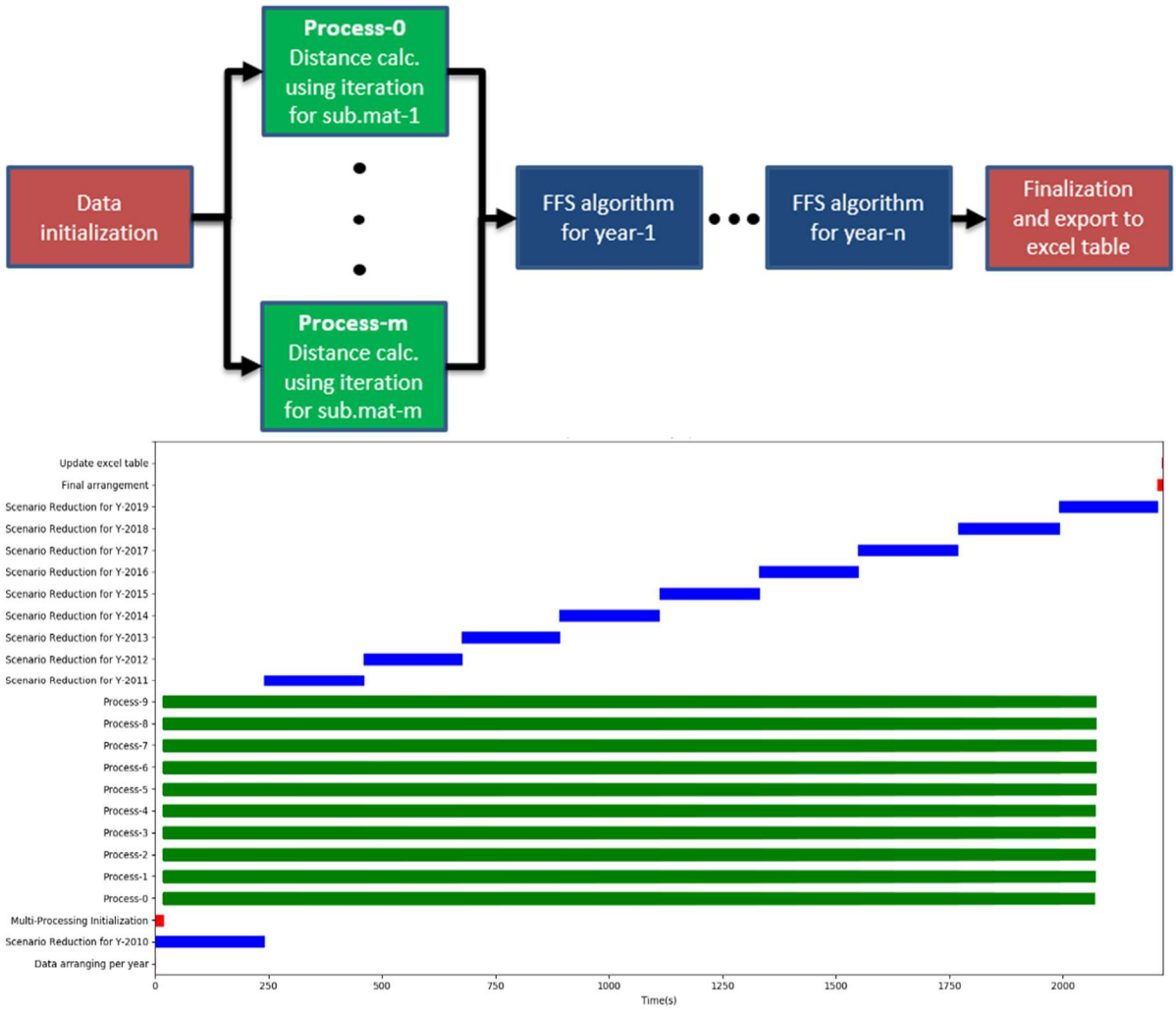

Fig.2 Multi-processing implementation for calculation of distance matrix, based on iteration

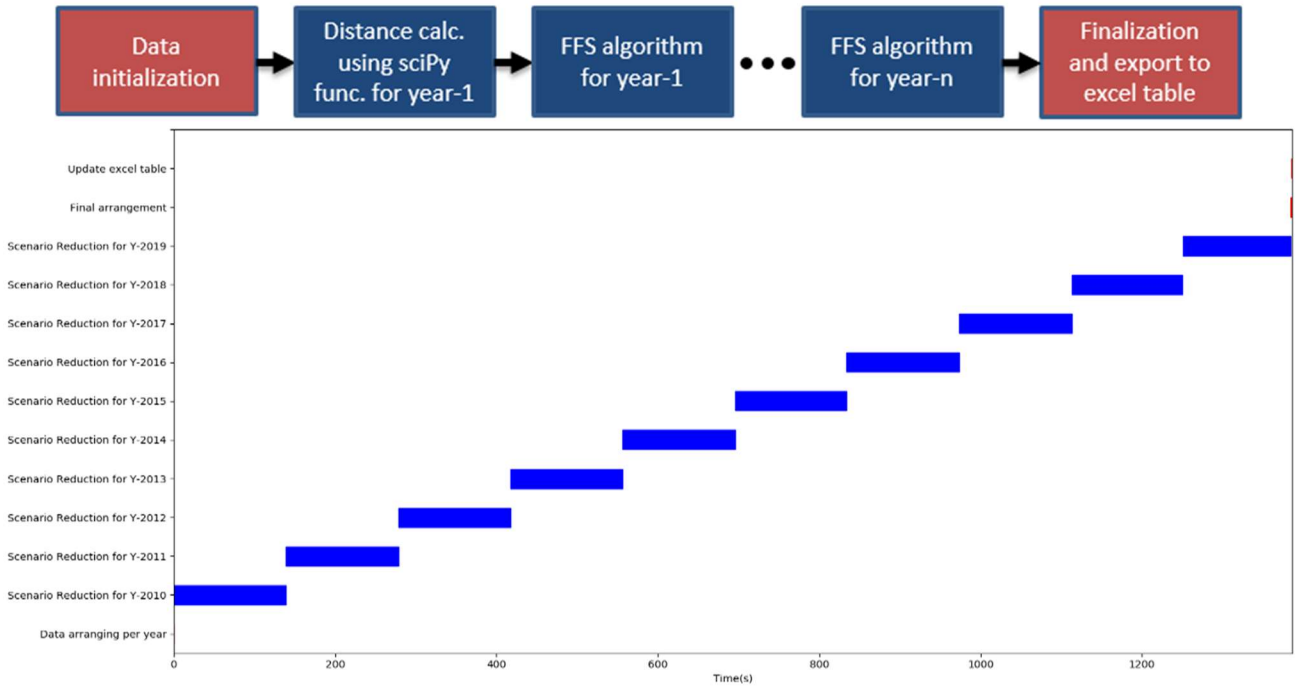

Fig.3 Single-processing implementation and sciPy-based distance matrix calculation

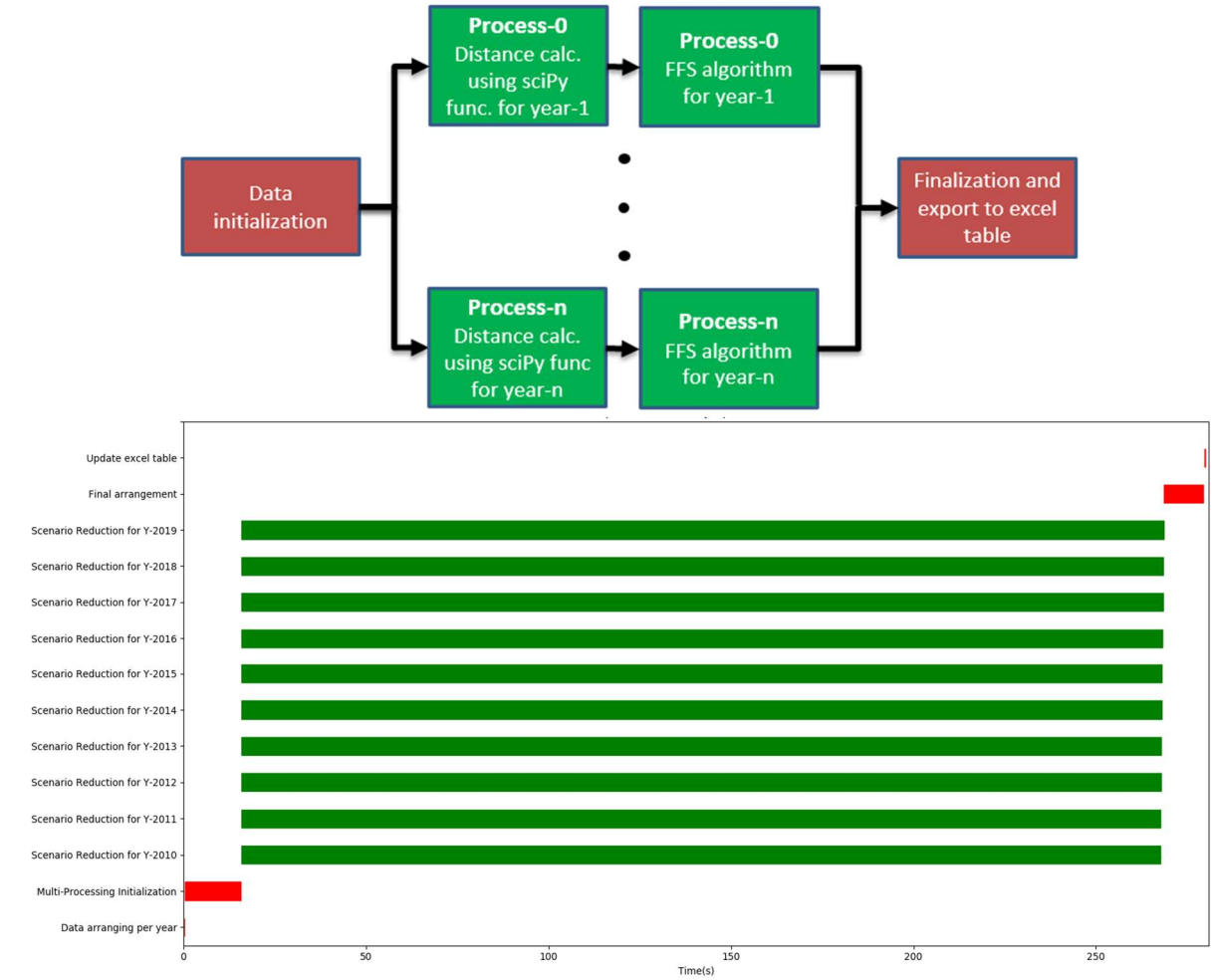

Fig.4 Multi-processing implementation for individual years and sciPy-based distance matrix calculation