**From:** Ivanov, Radoslav <ivanor@rpi.edu>   
**Sent:** Thursday, January 26, 2023 5:03 PM  
**To:** Ivanov, Radoslav <ivanor@rpi.edu>  
**Subject:** ICCPS RE Submission Instructions

Dear ICCPS authors,

Congratulations once again on being accepted to the ICCPS program!

We would like once again to encourage all authors of accepted papers to submit a repeatability evaluation package. The submission page is now live here: <https://iccpsre2023.hotcrp.com/>

As mentioned before, each package will get 2 reviews evaluating the following 3 criteria: 1) coverage, 2) documentation quality and 3) ease of reuse. To be accepted, your package will need to 1) reproduce at least 75% of all computational components; 2) be reasonably well documented so it's clear how to reproduce each component; 3) contain easy-to-run scripts for each component (and ideally be easily extended beyond the results in the paper). Full instructions can be found on the ICCPS website: <https://iccps.acm.org/2023/call-for-papers/index.html>

To ensure we can run your code, reviewers will have 1 week initially for a smoke test: they will try to make sure your code runs and sort out any installation/compilation issues. You will have 5 days after that to respond to any issues raised by the reviewers.

Important dates:

1) Submission deadline: Feb. 3 AoE

2) Smoke test due: Feb. 10

3) Author response to smoke test issues (if any) due: Feb. 15 AoE

4) Repeatability evaluation decisions: Mar. 6 (tentative)

Best Regards,

Rado and Daniel

**Repeatability Package Submissions Guideline**

Repeatability in cyber-physical systems research is critical. Therefore, this year we are strongly encouraging authors of **accepted regular papers** to submit a repeatability package. The program committee will evaluate this package. Authors of accepted repeatable packages will receive a repeatability badge that will be included on the first page of the published version. These papers will also be highlighted on the conference website. The submission date will be approximately a week after the notifications for the papers are sent out.

The *Repeatability Evaluation Package*(REP) consists of the following components:

* A **copy**(in pdf format) of the submitted paper with an appendix that explains the following:
  + What elements of the paper are included in the REP (e.g., figures, tables, etc.).
  + Instructions for installing the software.
  + Instructions for running the software. Ideally, there is a short and easy-to-run script for each computational component in the paper.
  + The system requirements for running the REP (e.g.: OS, compilers, environments, etc.). The document should also include a description of the host platform used to prepare and test the docker image or virtual machine.
  + Expected resource requirements. What architecture did you use for your experiments and how long is your code expected to run?
* The software. This should be made available with a link (for example, to GitHub or Google Drive; please don’t use your personal website) that should remain accessible throughout the review process. Please prepare either a:
  + Docker image (preferred).
  + **Virtual Machine**. You may use VirtualBox to save a VM image as an OVA file.
  + If your experiments use open-source simulators, please enclose them within the virtual machine or docker image.
  + If the previous options are not viable, please contact the RE PC chairs to make other arrangements. For example, if your software uses other licensed software (e.g., Matlab) which cannot be included in a VM or Docker image.
* Any data used in your experiments
  + Provide data as a tar.gz file with instructions for mounting it into the docker image or the virtual machine and passing the data as input to the code.
  + If the data is large, then please share it using an appropriate cloud storage solution. You mustn’t share any proprietary data that cannot be made open source.

Each REP needs to achieve satisfactory performance at the following three criteria in order to get the repeatability badge:

* **Coverage**. How many of the computational components in the paper can be reproduced?
* **Documentation quality**. Is the REP clearly documented, including system requirements, resource requirements, installation instructions and execution instructions?
* **Ease of reuse**. Is the REP generally easy to reuse, i.e., is the code well documented, are the scripts clear and easy to run?

Submission site: [https://iccpsre2023.hotcrp.com](https://iccpsre2023.hotcrp.com/)

Repeatability Evaluation Package (REP) Documentation for the Paper

“Learning Spatio-Temporal Aggregations for Large-Scale Capacity Expansion Problems”

# Overview

The methodology presented in the paper consists of two parts. The first part is the autoencoder (AE) procedure explained in Section 4, and the second part is the capacity expansion model (i.e., GTEP) presented in Section 2. Therefore, the REP consists of two main folder: 1)‘*AE-Module’* folder which contains all the data and codes used in the AE; 2) ‘*GTEP-Module’* folder which contains all the data and codes used in the GTEP.

Each successful run involves at most two steps. In the first step the AE module is run to obtain spatio-temporal aggregations for the network size and number of representative days. The second step uses the output from the first step to run the GTEP for the given parameters. Both modules are implemented in Python with transferability and legibility in mind. The details of both steps is given in the Section 3 and 4 of this document.

# System and Resource Requirements

All the codes are developed and tested in Python 3.8. The codes do not need installation as they are presented in the raw script format. The following packages are required for a successful run:

numpy, pandas, matplotlib, random, geopp, os, gurobipy, time, sys, pytorch, torch\_geometric, torch\_sparse, torch\_scatter, sklearn, scikit-learn-extra, networkx, tqdm, datetime

Note that the systems must have Anaconda as wells as Gurobi solver installed. We run the AE module on a personal MacBook laptop, and run the GTEP module on the MIT Supercloud system (Intel Xeon Platinum 8260 processor with up to 48 cores and 192 GB of RAM) with multiple instances running in parallel. However, both modules are successfully tested on Windows machines. All the packages are generic Python packages, so we expect a smooth run on other machines with recent version of Windows, Linux, or Mac.

# How to Run

This section explains the running procedure for both modules. The first module, which is AE, usually runs under 4 hours depending on the host machine and aggregation parameters. The run time of GTEP greatly varies based on the spatio-temporal aggregation level as well as problem parameters, but they are usually run under 10 hours.

## Autoencoder

As mentioned, all code and data related to the autoencoder are available in the folder ‘*AE*-*Module’*. The only code that needs to be run can be found in Autoencoder.ipynb, and all data is given in the ‘*Data’* folder.

To train the spatial and temporal aggregation autoencoders, the Jupyter notebook Autoencoder.ipynb can be run cell by cell. The data can be uploaded and processed by running the cells under the section “*Constructing Datasets*.” Specifically, the electricity load data is given in “Data/Power Network Topology-full network (188 nodes)/bus\_load\_RM\_2050.csv”, the NG data is given in “Data/ng\_daily\_load2050\_RM.csv”, the wind CF data is given in “Data/wind-CF-188-nodes.csv”, and the solar CF data is given in “Data/solar-CF-188-nodes.csv”.

To train the spatial aggregation autoencoder, one should run all cells in the section titled ‘*Spatial Aggregation Autoencoder*’. The parameters/variables to be tuned are as follows:

* Number of spatial clusters: the final spatial resolution (number of nodes) of the GTEP model. This tunes the graph pooling block in the autoencoder.
* Learning rate (*lr*): the learning rate for the Adam optimizer. This should be tuned to minimize the autoencoder validation loss.
* Epochs: the number of iterations of gradient descent during training. This should also be tuned to minimize the autoencoder validation loss.

The code then saves the learned spatial clusters to the file ‘spatial\_cluster.csv’. This file has two columns:

* Node: the node label.
* Cluster: the assigned cluster of the corresponding node, or in other words, the node label to which it is absorbed in the spatially aggregated GTEP.

Similarly, to train the temporal aggregation autoencoder, one should run all cells before the Training subsection under ‘*Spatial Aggregation Autoencoder’* as well as all cells in the section titled ‘*Temporal Aggregation Autoencoder’*. The parameters/variables to be tuned are as follows:

* Days: the number of representative days to be used in the temporally aggregated GTEP. This is a hyperparameter for the K-medoids algorithm.
* Learning rate (*lr*): the learning rate for the Adam optimizer. This should be tuned to minimize the autoencoder validation loss.
* Epochs: the number of iterations of gradient descent during training. This should also be tuned to minimize the autoencoder validation loss.
* *alpha\_G, alpha\_W, alpha\_S*: the objective weight coefficients corresponding to NG, wind CF, and solar CF reconstruction loss (relative to the electricity reconstruction loss).
* *max\_iter*: the number of iterations for the K-medoids algorithm.

The code then saves the learned representative days to the file temporal\_cluster.csv. This file has three columns:

* Day of Year: number corresponding to the day of the year (e.g., 0 corresponds to January 1)
* Date
* Weight: the relative weight of the representative day in the temporally aggregated GTEP. This is given by the number of members of the cluster as learned by the K-medoids algorithm.

## Running GTEP

Once the Autoencoder codes are run, the output is stored in ‘*joint\_CF\_with\_extreme\_days’* folder insider the ‘GTEP-Module’ folder. The ‘GTEP-Module’ folder contains several folders, Python scripts, and CSV file from which we only explain the files that are directly related to running the GTEP model. The GTEP model runs from UB.py script. The code can both be run from the Anaconda’s command window or an IDE such as Spyder. The scrips require 5 key parameter as follows:

* Network size: the number of nodes in the power system. The parameters should set to one 6, 10, 15, or 20. Default value is 6.
* Number of representative days: the number of planning days considered in the model. The parameters can take any integer number between 2 and 30. Default value is 3.
* Solver Gap: the MIP gap for Gurobi to terminate. The parameters is any number between 0 and 1. Default value is 0.01.
* Solver Time Limit in terms of hours: the solution time limit for the solver. The parameter can take any integer number. Default value is 1.
* Solver Thread: the number of CPU core threads dedicated to the solver. The parameters takes integer numbers and can vary between 1 and the number of thread on the machine. Default is 4.

The following shows these parameters in the UB.py opened in Spyder:

Text

Description automatically generated

The following is an example of running the script from the anaconda prompt:

Text

Description automatically generated

The parameters after UB.py should be separated by a comma and are in the order they were explained above.

The results of each run is stored in the output file JPoNG\_Results.csv. The code reports a detailed output in 87 columns and two rows. The first row is the header for each column and the other column is the value for each column. The initial columns show the parameters of the instance. Column K and L show MIP gap and run time of the problem in seconds. Other columns provide details of the solution and are named mnemonically for readability. The following columns used in the paper:

* Column N (Total-cost): total cost of the GTEP model including the total cost of power and NG systems
* Column O (*Power-cost*): total cost of the power system
* Column P (*est-cost*): power plant establishment cost
* Column R (*FOM*): fixed operating and maintenance cost for the power system
* Column S (*VOM*): variable operating and maintenance cost for the power system
* Column AH (*NG*-*cost*): total cost of the NG system

The following figure shows part of the solution output for an instance:



All computational experiments reported in the paper can be repeated by running the GTEP model with different parameters.