This tutorial goes through the steps of downloading the reference GAMS code, explains the purpose of each component, how to make it accessible on GitHub, how to make a submission, how to download the results, and how to interpret the contents of the zip file.

The GAMS example

The General Algebraic Modeling System (GAMS) is a commercial tool used to solve a wide variety of optimization problems, including ACOPF. A set of example GAMS scripts is available on GitHub. This repository contains the following required scripts (the other files in the repository are not necessary for this example):

1. My Gams1.gms
2. My Gams2.gms
3. pscopf.gms
4. pscopf_compute_solution.gms
5. pscopf_prepare_data.gms
6. pscopf_process_solution.gms
7. pscopf_run.gms
8. pscopf_write_solution.gms

The functionality of each of these GAMS scripts is as follows:

MyGams1.gms
This is the main script invoked during the first (timed) evaluation. It takes a single parameter (case) that is the name of the input data file (pscopf_data.gms) in the directory for the scenario being evaluated. The script invokes pscopf_run.gms, which invokes additional scripts to prepare the data, generate the power flow solutions for base case and contingency cases, and create the required output files solution1.txt and solution2.txt. Solution0.txt, containing the values of every variable in the model and every constraint violation, and solution3.txt, containing the power balance constraint violation values (in physical units, not p.u.), will also be created. For the purposes of the competition, only solution1.txt and solution2.txt are needed. However, solution0.txt is useful for debugging and for really understanding the model. Solution3.txt is useful for understanding the source of feasibility violations. This script also specifies which nonlinear programming solver (knitro is the default) will be used.

MyGams2.gms
This is the main script invoked during the second (untimed) evaluation. For this example, MyGams2.gms is functionally the same as MyGams1.gms, but all GAMS submissions must have a MyGams2.gms. In principle the solution values for solution1.txt are available much sooner than those for solution2.txt. This is the purpose of splitting the GAMS model into MyGams1.gms and MyGams2.gms, however, this example GAMS model solves the whole model and produces both solution values at the same time. If

the evaluation protocol detects the presence of solution2.txt after MyGams1.gms is executed, it skips the execution of MyGams2.gms, but this information is not available when the scripts are downloaded so an error message will result if MyGams2 is missing:

Status	Status Notes
error	No MyGams2.gms at /home/svcarpacomp/submission-manager-tmp/UserID/SubmissionID/MyGams2.gms. All GAMS submissions must contain a MyGams2.gms.

For MyGams2.gms to be executed by the evaluation platform, this example MyGams1.gms would have to be modified to not produce a solution2.txt file.

pscopf_run.gms
This script invokes pscopf_prepare_data.gms and then pscopf.gms, which runs the whole model.

pscopf_prepare_data.gms
This script reads the input data from pscopf_data.gms and converts it to a binary gdx file for use by pscopf.gms

pscopf.gms
This script is the main model code. It declares the data parameters, variables, constraints, and models, then loads the data from the gdx file created by pscopf_prepare_data.gms and invokes the chosen solver to solve the model. The *feastol* variable is set to 1e-8 in this script (line 582); increasing it may result in infeasible solutions—with *feastol* set to 1e-6, the Phase 0 IEEE 14 Bus dataset has 6 infeasible scenarios (nos. 17, 22, 35, 67, 85, and 99) and the Phase 0 Modified IEEE 14 Bus dataset has 8 infeasible scenarios (nos. 2, 47, 49, 53, 59, 76, 84, and 99). The script can fix some variables to their bounds if needed and resolve. Certain post solve quantities can also be calculated, such as constraint violations, by pscopf_compute_solution.gms. Before exiting the script calls pscopf_write_solution.gms three times to produce the files solution0.txt, solution1.txt, and solution2.txt.

pscopf_compute_solution.gms
This script computes the solution from variable values as in the evaluation code and writes the file solution3.txt containing the power balance constraint violation values (in physical units, not p.u.). Not invoking this script in pscopf.gms simply prevents the generation of solution3.txt, which is not necessary for GO Competition evaluation.

pscopf_write_solution.gms
This script writes the computed solution to a file. Depending on the value the parameter “outputtype”, this file can write the solution in three different formats:

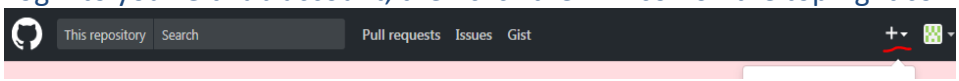
1. solution1.txt containing just the values of generator output in the base case;
2. solution2.txt containing the values of the other variables needed to fully evaluate the solution, i.e. contingency cases;
3. solution0.txt containing the values of every variable in the model and every constraint violation. For the purposes of the competition, only solution1.txt and solution2.txt are needed. However solution0.txt is useful for debugging and for really understanding the model.

pscopf_process_solution.gms
This script evaluates the maximum constraint violations; used by pscopf_write_solution.gms.

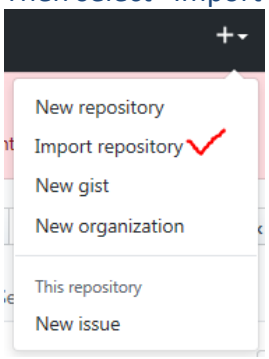
Putting GAMS in GitHub

After creating your own GitHub account, there are two ways to place the GAMS example in your account:

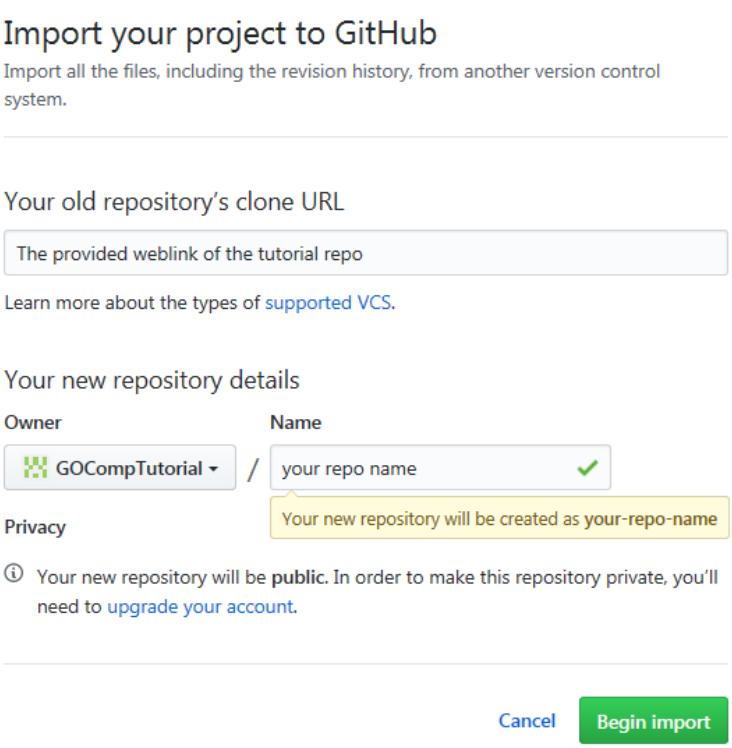
1. Create a new repo by importing from the given link;
   a. Log into your GitHub account, then click the "+" icon on the top right corner,



   b. Then select "Import repository",

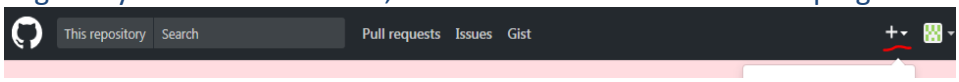


   c. Fill in the proper importing information, and click "Begin import"

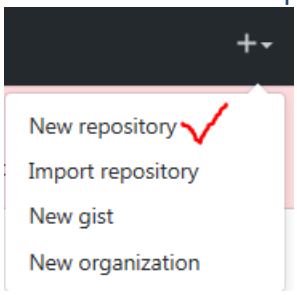


d. When the import is complete all the files for the GAMS example will be in your repository.

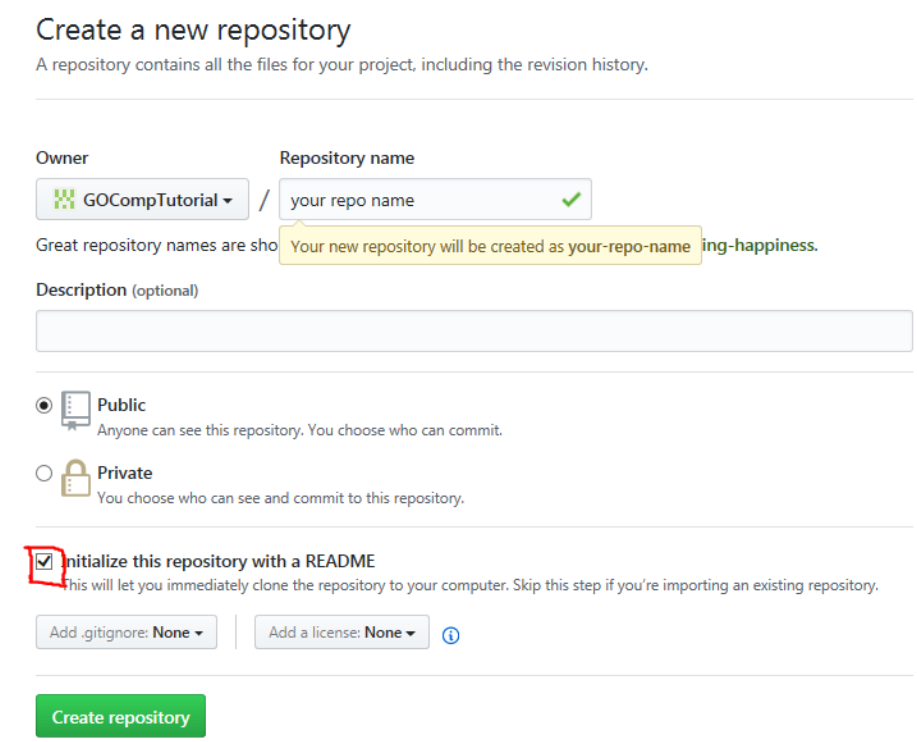
2. Create a new repository by uploading from your local machine with the downloaded files
   a. Log into your GitHub account, then click the "+" icon on the top right corner,



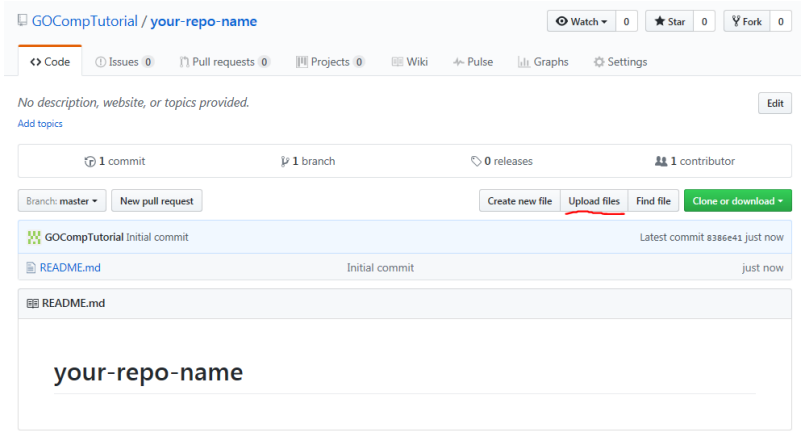
   b. Then choose "New repository",



   c. Fill in the necessary repository information. The Repository name chosen here will be needed when making a GO Competition submission. The Repository name can be accessed by the URL https://github.com/your_user_name/your-repo-name/



d. Click the green “Create repository” button and “Upload files” (next screen) will provide you with the option to select local files to upload.

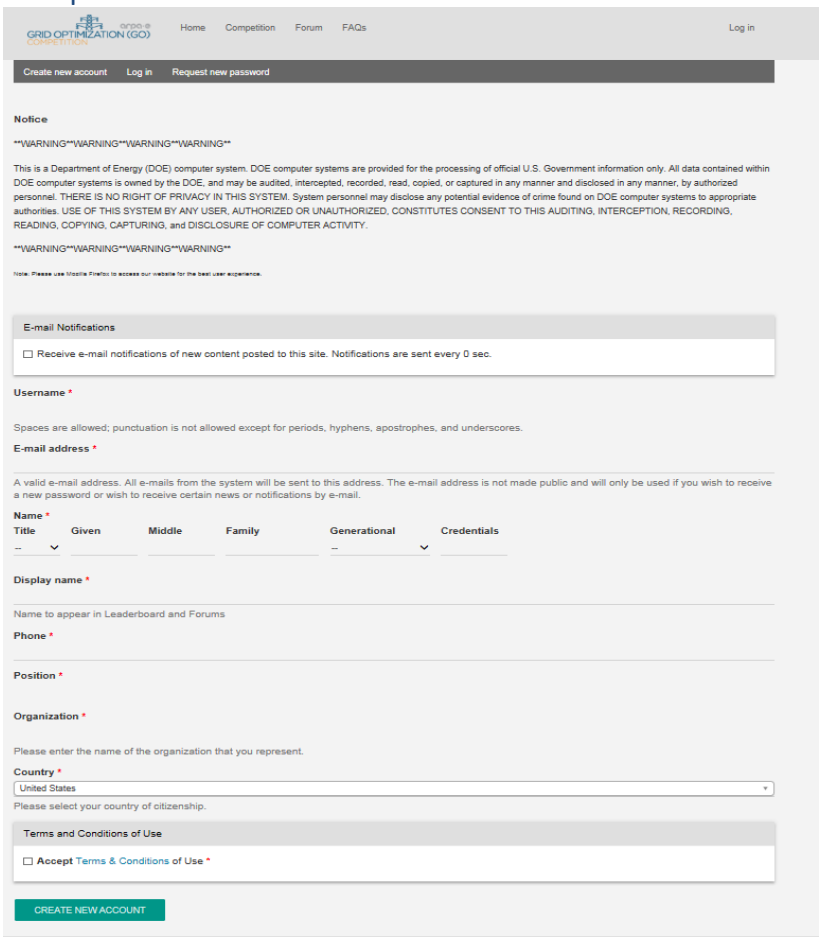


e. GitHub allows you to create branches from this repository, starting with the Master branch. During the submission process you will be asked if you wish to use a branch other than the Master branch, which is the default.

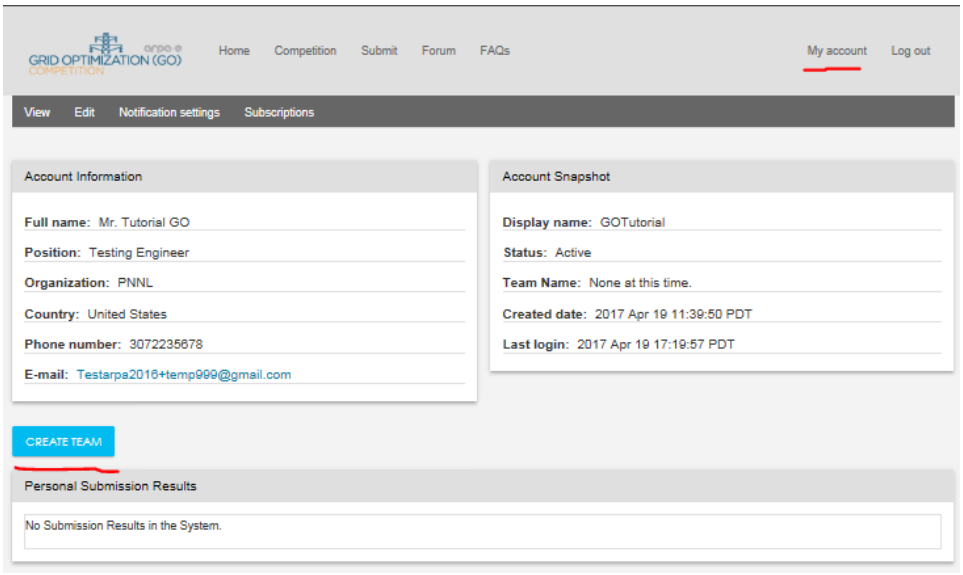
Create a GO Competition account

Now you have set up a GitHub repository that hosts a GAMS example, you are ready to complete the GO Competition registration steps before making a submission.

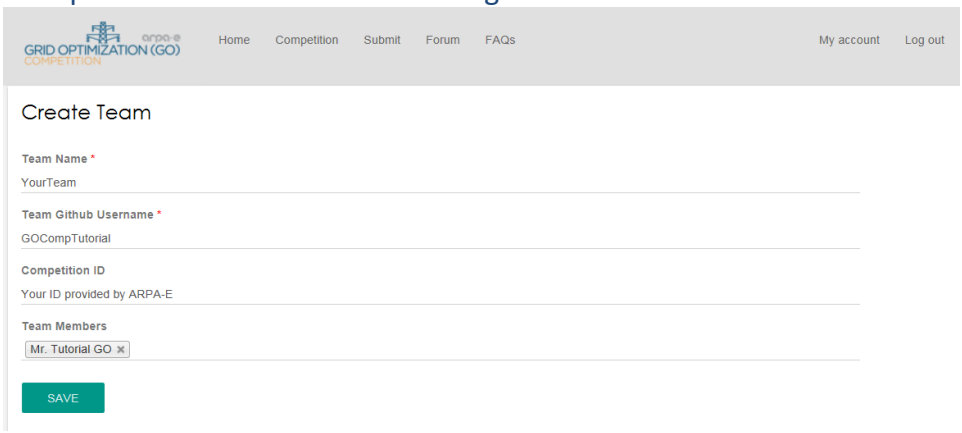
1. Register as a new entrant on Grid Optimization Competition website that can be accessed from the “Log in” link in the upper right of the GO Competition home page and the “Create new account” link in the upper left of the login page (or directly at https://goccompetition.energy.gov/user/register)
2. Fill in the required information



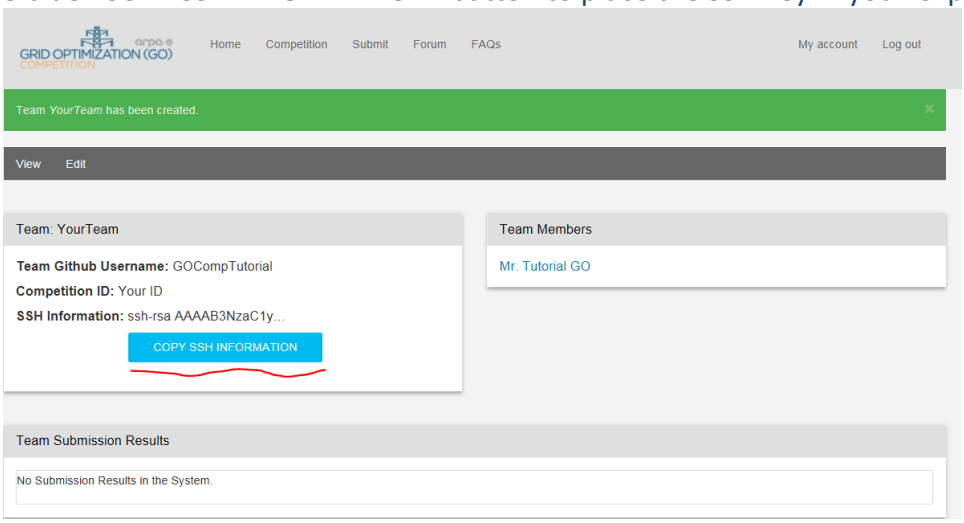
3. Click the green “Create New Account” button at the bottom



4. Click the blue “Create a Team” button page (even if just one person) on the resulting My account page
5. Fill in the required information and click the green “Save” button.



6. Click the blue “COPY SSH INFORMATION” button to place the SSH key in your Clipboard.



7. Use this SSH key to configure your GitHub account so GO Competition can access it and read the example GAMS scripts.

Submitting the GAMS example

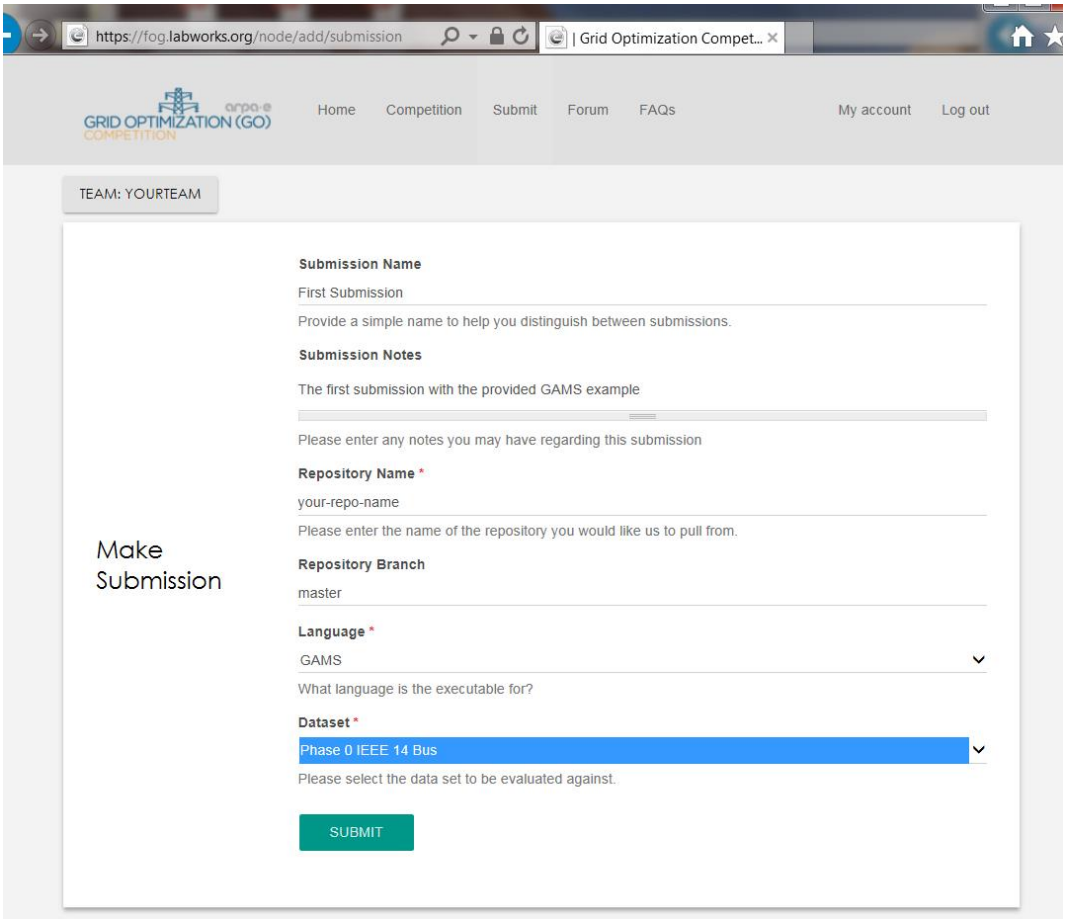
Click the "Submit" tag at the top of any page (it is only visible after you have registered and logged in).

Complete the submission form with required information (red asterisk):

- Repository Name (the name you chose is step 2d under "Create a GO Competition account"
- Language (chose form the dropdown menu; GAMS for this example)
- Dataset ((chose form the dropdown menu; IEEE 14 Bus for this example)

and other information relevant to you:

- Submission Name (a simple name to distinguish between submissions)
- Submission Notes (relevant information you want to associate with this submission)
- Repository Branch (the default is master; use this unless you want to use another branch you created)



Your submission is complete when you click on the green "Submit" button.

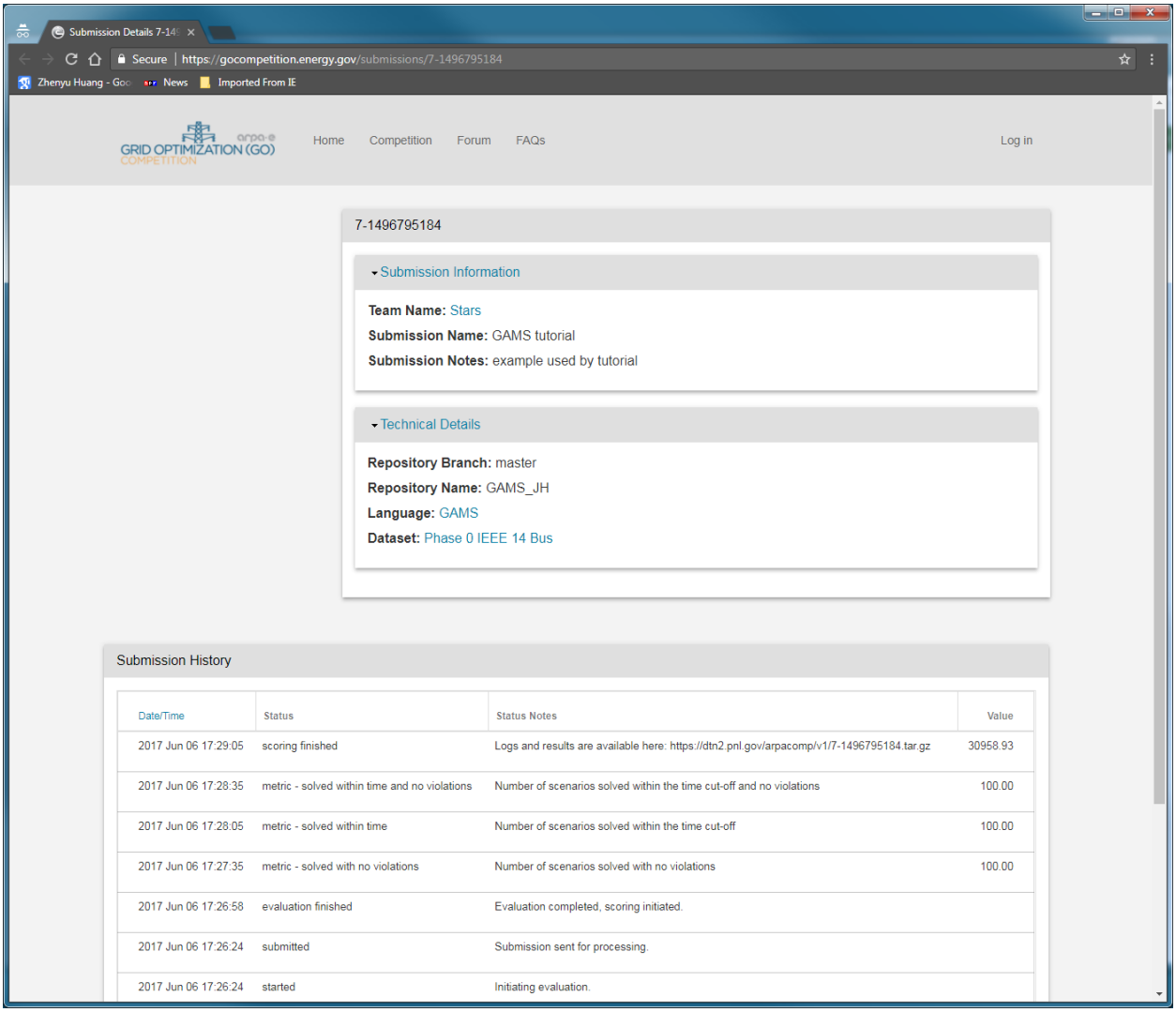
Once you make the submission, it will automatically be placed in the processing queue and you will be taken to the status page for your submission that now has a unique submission ID shown in the first section of the page. The Submission History section will say “No Submission Results in the System.” This is because the Evaluation Platform has not had time to respond before this page was created. If you refresh the page you will see the date and time your submission was made and that it was sent for processing. Throughout the course of the evaluation process this page will be updated with status messages that also include started, evaluation finished, metrics (how many scenarios have no time or constraint violations or either), and scoring finished.

If the Status message says “error” and the Status Notes say “Error cloning Git repository” it probably means you have not installed your SSH key.

Access to the submission pages for all your submissions is available by selecting “My account” on the top of any page if you are logged in or by selecting the Team button below the Leaderboard summary on the left side of any of the Competition pages

All the submissions are processed based on the order received.

After your submission is processed, the computation results and log files will be zipped into a single file for downloading. The URL for this file is given in the “scoring finished” Status Notes.



Unzipping and extracting the .tar.gz file is system dependent. For Linux the command is
tar -xzf submissionID.tar.gz
where submissionID is the numeric string before .tar.gz.
For Windows open source software such as 7-Zip or commercial software such as WinZip can be used.

Within the zipped file, there is a scenario_results.csv file, a score.txt file and one sub-folder for each scenario of the selected dataset.

The scenario_results.csv file contains the parameters used to score the submission (lines 1-10) and, for each scenario, the official time (seconds). There may also be language dependent information columns. For the reference GAMS submission the additional columns include the (third) line of each solution0.txt that reports the time (seconds), objective function value (dollars), the maximum constraint violation, and the GAMS Solver and Model Status Codes.

Some of the Solver Status Codes are 1=Normal Completion; 2=Iteration Interrupt, i.e., reached iteration limit, 3=Resource Interrupt, i.e., reached time limit; and 4=Terminated by Solver, i.e., failed to converge. Some of the Model Status Codes are 1=Optimal; 2=Locally Optimal; 3=Unbounded; 4=Infeasible; 5=Locally Infeasible; 6=Intermediate Infeasible; and 7=Intermediate Nonoptimal.

The first and last few lines of an example scenario_results.csv are:

submission_date,Thu May 11 20:24:48 UTC 2017
submission_id,7-1494534246
Data set,Phase_0_Modified_IEEE14
Nominal time,1.
Nominal objective,90000.
Time scale,5
Constraint violation penalty scale,1000
Time violation penalty scale,100
Max infeasibility,1e-6
Number of scenarios,100
scenario_#,elapsed_seconds,gms_seconds,gms_objective_value,gms_constrViolMax,gms_solveStat,gms_modelStat
1, 0.256449031 ,0.1829997869,14678.2205765948,0.0000000000,1,2
2, 0.249230254 ,0.1929996070,16884.0158014400,0.0000000000,1,2
3, 0.197696185 ,0.1460003899,20986.5556790469,0.0000000000,1,2
4, 0.276676696 ,0.2099999925,34632.0470395088,0.0000000000,1,2
…
96, 0.249430824 ,0.1899997238,15636.3118932237,0.0000000000,1,2
97, 0.241333647 ,0.1789997332,24608.5793078142,0.0000000000,1,2
98, 0.251331858 ,0.1920000650,51848.7973899003,0.0000000000,1,2
99, 0.253997894 ,0.1960001187,52087.1337544410,0.0000000000,1,2
100, 0.272695136 ,0.2050003968,17448.8183514392,0.0000000000,1,2

The submission_date gives the Universal Coordinated (UTC aka Greenwich Time) when the submission was made. The submission_id is a unique label for this submission. The Data set is the name of the dataset chosen for the evaluation. The nominal time is the largest time in seconds taken by any of the scenarios during benchmarking and rounded up. The nominal objective is the largest objective value of any of the scenarios, also rounded up. It is the basis of the penalty function. The time scale is the factor used to establish the time threshold when a penalty is added. The time threshold is the product of the Nominal time and the Time scale; in this case 5, i.e., any scenario that takes more than 5 seconds is penalized. The Constraint violation penalty scale is the factor used in creating the constraint violation penalty, i.e., any base case or contingency with a feasibility greater than the feasibility threshold is penalized by adding the product of the Nominal objective and the Constraint violation penalty scale (in this case 9e+7) to the scenario score. The Time violation penalty scale does the same for scenarios that exceed the time threshold, i.e., the product of the Nominal objective and the Time

violation penalty scale (in this case 9e+6) is added to the scenario score. The scenario score without penalties is simply the objective function value so lower is better. The Max infeasibility is the feasibility threshold, i.e., a feasibility larger than this value incurs a penalty. Finally, Number of scenarios gives the number of scenarios used in this evaluation.

The score.txt file contains the same submission ID, time and dataset information as the contingency file and also contains information used to score the results of each scenario. This information includes the scenario score, the objective value determined by the scoring algorithm (not the GAMS determined value), the maximum violation, the contingency associated with the maximum violation (0 is the base case), and the computation time (the same value that appears in scenario_results.csv. If the maximum violation is less than the Max infeasibility value in scenario_results.csv and the computation time is less than the product of the nominal time and the time scale, also from scenario_results.csv, then the score is equal to the objective value. The constraint and time penalties, if appropriate, are applied as explained above.

Each scenario sub-folder contains the “solution1.txt” and “solution2.txt” files used for scoring. Additional language dependent results and log files are also included. For the reference GAMS submission the additional files are log1.txt, MyGams1.lst, pscopf.lst, pscopf_prepare_data,lst, pscopf_run.lst, solution0.txt, solution3.txt and submission.log.