Tutorial for submitting jobs using the JobManager

Zhongnan Xu

2013-02-05 Tue

Contents

1 Introduction

The purpose of the `JobManager` module is to provide a python wrapper for submitting and tracking jobs in a queue environment.

2 Configuration

The `JobManager` is initially built for a PBS queue environment, so many of the commands will have to be modified for usage in a different queue environment. These customizations will likely take place in the following files.

1. The `submit` and `write_submit` function in the `structopt/utilities/job_manager.py` file will likely need to be updated to reflect your specific queue environment.
2. The dictionaries held in `structopt.utilities/rc.py` is the first attempt to store some dictionaries specific to the queue environment. Many queue specific variables are drawn from here.

3 Submitting jobs

3.1 Single job

The script below is an example script of submitting a single job to a queue using the $\mathbf{job}_{\mathbf{manager.py}}$. The optimization run is a short run of a Au55 nanoparticle using only LAMMPS. A large part of the script is defining the input, which goes into the `Job` class. These inputs are given below.

1. `calcdir`: This is a string that tells where the calculation is run. Note that the calculation itself is run within the **calcdir/logs{time}** directory, which is created when the job starts to run on the queue. Unless an absolute path, the **calcdir** directory is always given with respect to directory that the job script is run from

2. `optimizer`: This is a string of the optimizer file used for the calculation. These files can be found in the **structopt/optimizers** folder. Upon run, a copy of this script is placed insde of the **calcdir** directory and run from there.

3. `StructOpt_parameters`: This is a dictionary object that should mirror the input file you are trying to submit

4. `submit_parameters`: This dictionary holds the submit parameters. These will be specific to the queue system in use. In this example, we specify the the submission system, queue, number of nodes, number of cores, and walltime.

```
from structopt.utilities.job_manager import Job
from structopt.utilities.exceptions import Running, Submitted, Queued

calcdir = 'job_manager_examples/Au55-example'

LAMMPS_parameters = {"use_mpi4py": True,
                     "MPMD": 0,
                     "keep_files": False,
                     "min_style": "cg",
                    "min_modify": "line quadratic",
                     "minimize": "1e-8 1e-8 5000 10000",
                     "pair_style": "eam",
                     "potential_file": "$STRUCTOPT_HOME/potentials/Au_u3.eam",
                     "thermo_steps": 0}

StructOpt_parameters = {
    "seed": 0,
    "structure_type": "cluster",
    "generators": {"sphere": {"number_of_individuals": 20,
                              "kwargs": {"atomlist": [["Au", 55]],
                                         "cell": [20, 20, 20]}}},
    "fitnesses": {"LAMMPS": {"weight": 1.0,
                   "kwargs": LAMMPS_parameters}},
    "relaxations": {"LAMMPS": {"order": 0,
                               "kwargs": LAMMPS_parameters}},
    "convergence": {"max_generations": 10},
    "mutations": {"move_atoms": {"probability": 0.1},
                  "rotate_cluster": {"probability": 0.1}},
    "crossovers": {"rotate": {"probability": 0.7}},
    "predators": {"best": {"probability": 1.0}},
    "selections": {"rank": {"probability": 1.0,
                            "kwargs": {"unique_pairs": False,
                                       "unique_parents": False}}},
    "fingerprinters": {"keep_best": True,
                       "diversify_module": {"probability": 1.0,
                                            "kwargs": {"module": "LAMMPS",
                                                       "min_diff": 0.001}}},
    "post_processing": {"XYZs": -1},
}

submit_parameters = {'system': 'PBS',
                     'queue': 'morgan2',
                     'nodes': 1,
                     'cores': 12,
                     'walltime': 12}

optimizer = 'genetic.py'
```

```
job = Job(calcdir, optimizer, StructOpt_parameters, submit_parameters)
job.optimize()
```

Upon running this script, the user should get back an exception called `structopt.utilities.exceptions.Subm` with the jobid. This is normal behavior and communicates that the job has successfully been submitted.

3.2 Multiple jobs

One advantage of the job manager is that it allows one to submit multiple jobs to the queue. This is often useful for tuning the optimizer against different inputs. The script below is an example of submitting the same job at different seeds.

In the previous script, submitting a single job successfully with `Job.optimizer` method resulted in an exception. We can catch this exception with a **try** and **except** statement. This is shown below in the script where upon a successful submission, the script prints out the jobid to the user.

```
from structopt.utilities.job_manager import Job
from structopt.utilities.exceptions import Running, Submitted, Queued

LAMMPS_parameters = {"use_mpi4py": True,
                     "MPMD": 0,
                     "keep_files": False,
                     "min_style": "cg",
                    "min_modify": "line quadratic",
                     "minimize": "1e-8 1e-8 5000 10000",
                     "pair_style": "eam",
                     "potential_file": "$STRUCTOPT_HOME/potentials/Au_u3.eam",
                     "thermo_steps": 0}

StructOpt_parameters = {
    "seed": 0,
    "structure_type": "cluster",
    "generators": {"sphere": {"number_of_individuals": 20,
                              "kwargs": {"atomlist": [["Au", 55]],
                                         "cell": [20, 20, 20]}}},
    "fitnesses": {"LAMMPS": {"weight": 1.0,
                   "kwargs": LAMMPS_parameters}},
    "relaxations": {"LAMMPS": {"order": 0,
                               "kwargs": LAMMPS_parameters}},
    "convergence": {"max_generations": 10},
    "mutations": {"move_atoms": {"probability": 0.1},
                  "rotate_cluster": {"probability": 0.1}},
    "crossovers": {"rotate": {"probability": 0.7}},
    "predators": {"best": {"probability": 1.0}},
    "selections": {"rank": {"probability": 1.0,
                            "kwargs": {"unique_pairs": False,
                                       "unique_parents": False}}},
    "fingerprinters": {"keep_best": True,
                       "diversify_module": {"probability": 1.0,
                                            "kwargs": {"module": "LAMMPS",
                                                       "min_diff": 0.001}}},
    "post_processing": {"XYZs": -1},
}

submit_parameters = {'system': 'PBS',
                     'queue': 'morgan2',
                     'nodes': 1,
                     'cores': 12,
                     'walltime': 12}

optimizer = 'genetic.py'
```

```
seeds = [0, 1, 2, 3, 4]
for seed in seeds:
    StructOpt_parameters['seed'] = seed
    calcdir = 'job_manager_examples/Au55-seed-{}'.format(seed)

    job = Job(calcdir, optimizer, StructOpt_parameters, submit_parameters)

    try:
        job.optimize()
    except Submitted:
        print(calcdir, job.get_jobid(), 'submitted')
```

4 Tracking jobs

In the previous section, we covered how to submit a new job from an empty directory. This is done by first initializing an instance of the `StructOpt.utilities.job_manager.Job` class with a calculation directory along with some input files and then submitting the job with the `Job.optimize` method. The `Job.optimize` method knows what to do because upon initialization, it detected an empty directory. If the directory was not empty and contained a StructOpt job, the $\text{job}_{\text{manager}}$ knows what to do with it if `Job.optimize` was run again. This is all done with exceptions.

The three primary exceptions that are returned upon executing the `Job.optimize` method are below along with their reasoning.

1. `Submitted`: This exception is returned if a job is submitted from the directory. This is done when `Job.optimize` is called in an empty directory or `Job.optimize` is called with the kwarg `restart=True` in a directory that is not `Queued` or `Running`.

2. `Queued`: The job is queued and has not started running. There should be no output files to be analyzed.

3. `Running`: The job is running and output files should be continously be updated. These output files can be used for analysis before the job has finished running.

4. `UnknownState`: This is returned if the `calcdir` is not an empty directory doesn't detect it as a StructOpt run.

Note that if no exception is returned, it means the job is done and is ready to be analyzed. `Job.optimize` does nothing in this case.

One way of using these three exceptions is below. If the job is submitted or Queued, we want the script to stop and not submit the job. If it is running, additional commands can be used to track the progress of the job. This is done through the `DataExplorer` module.

```
from structopt.utilities.job_manager import Job
from structopt.utilities.exceptions import Running, Submitted, Queued

calcdir = 'job_manager_examples/Au55-example'

LAMMPS_parameters = {"use_mpi4py": True,
                     "MPMD": 0,
                     "keep_files": False,
                     "min_style": "cg",
                    "min_modify": "line quadratic",
                     "minimize": "1e-8 1e-8 5000 10000",
                     "pair_style": "eam",
```

```
                    "potential_file": "$STRUCTOPT_HOME/potentials/Au_u3.eam",
                    "thermo_steps": 0}

StructOpt_parameters = {
    "seed": 0,
    "structure_type": "cluster",
    "generators": {"sphere": {"number_of_individuals": 20,
                              "kwargs": {"atomlist": [["Au", 55]],
                                         "cell": [20, 20, 20]}}},
    "fitnesses": {"LAMMPS": {"weight": 1.0,
                  "kwargs": LAMMPS_parameters}},
    "relaxations": {"LAMMPS": {"order": 0,
                               "kwargs": LAMMPS_parameters}},
    "convergence": {"max_generations": 10},
    "mutations": {"move_atoms": {"probability": 0.1},
                  "rotate_cluster": {"probability": 0.1}},
    "crossovers": {"rotate": {"probability": 0.7}},
    "predators": {"best": {"probability": 1.0}},
    "selections": {"rank": {"probability": 1.0,
                            "kwargs": {"unique_pairs": False,
                                       "unique_parents": False}}},
    "fingerprinters": {"keep_best": True,
                       "diversify_module": {"probability": 1.0,
                                            "kwargs": {"module": "LAMMPS",
                                                       "min_diff": 0.001}}},
    "post_processing": {"XYZs": -1},
}

submit_parameters = {'system': 'PBS',
                     'queue': 'morgan2',
                     'nodes': 1,
                     'cores': 12,
                     'walltime': 12}

optimizer = 'genetic.py'

job = Job(calcdir, optimizer, StructOpt_parameters, submit_parameters)
try:
    job.optimize()
except (Submitted, Queued):
    print(calcdir, job.get_jobid(), 'submitted or queued')
except Running:
    pass
```

5 TODO Integration with the DataExplorer