Report 08/02/2021

Alessandro Franca

Pod_Aggregator_v9

Directory *Baseline:*
Contiene le nuove baseline prese da Libra.
local_opt_result.py:
Necessaria per separare la fase di ottimizzazione locale da quella di aggregazione (vedi file *app.py* contenente le APIs):

- La API *local_optimization* che effettua le ottimizzazioni locali ritornerà alla webapp chiamante un oggetto LocalOptimizationResult (serializzato json)
- Quando la webapp richiederà l'aggregazione, passerà l'oggetto LocalOptimizationResult serializzato alla API *aggregate*, che lo deserializzerà, e creerà l'aggregatore a partire dall' oggetto LocalOptimizationResult.

aggr_main.py:
Test di nuove configurazioni per Pod misti.
Ottimizzazioni Locali e Aggregazione con le nuove configurazioni miste di prova.

basic_profile.py:
Classe aggiunta per rappresentare i componenti BESS1, BESS2, PONTLAB1, PONTLAB2

aggregator.py:
Contiene la classe Aggregator, utilizzata per implementare l'aggregatore.
Principali attributi:
self.pods: pod presenti nell'aggregatore
self.local_optimization_result: istanza di *LocalOptimizationResult,* creato per poter effettuare l'aggregazione, a partire da un insieme di pod gia ottimizzati localmente (necessario ai fini di separare nelle APIs la fase di ottimizzazione locale, da quella di aggregazione)

Principali metodi:

- *resolve_pods(self):* ottimizza localmente i pod presenti nell'aggregatore (se devono essere ottimizzati)
- *resolve_pods_multiprocessing(self)*: ottimizza i pod presenti nell'aggregatore in parallelo.
- *resolve_pods_and_aggregate(self):* effettua direttamente sia la fase di ottimizzazione locale, che quella di aggregazione

Metodi aggiunti/aggiornati con le ultime modifiche:

- *aggregate(self):* effettua l'aggregazione. Se è presente un'istanza di *LocaloOptimizationResult* inizializza l'aggregatore a partire da quella, altrimenti la crea con la lista di Pod che possiede.
  Inizializza l'input per il Solver come segue:
  [SCREEN]

self.input['minimized']['flexibilities'] -> array di tutte le flessibilità minime dei pod dell'aggregatore (idem per maximized)
self.input['buy']['cost'] -> costo di acquisto di energia dalla grid per ogni istante di tempo
self.input['sell']['cost'] -> costo di vendita di energia alla grid per ogni istante di tempo

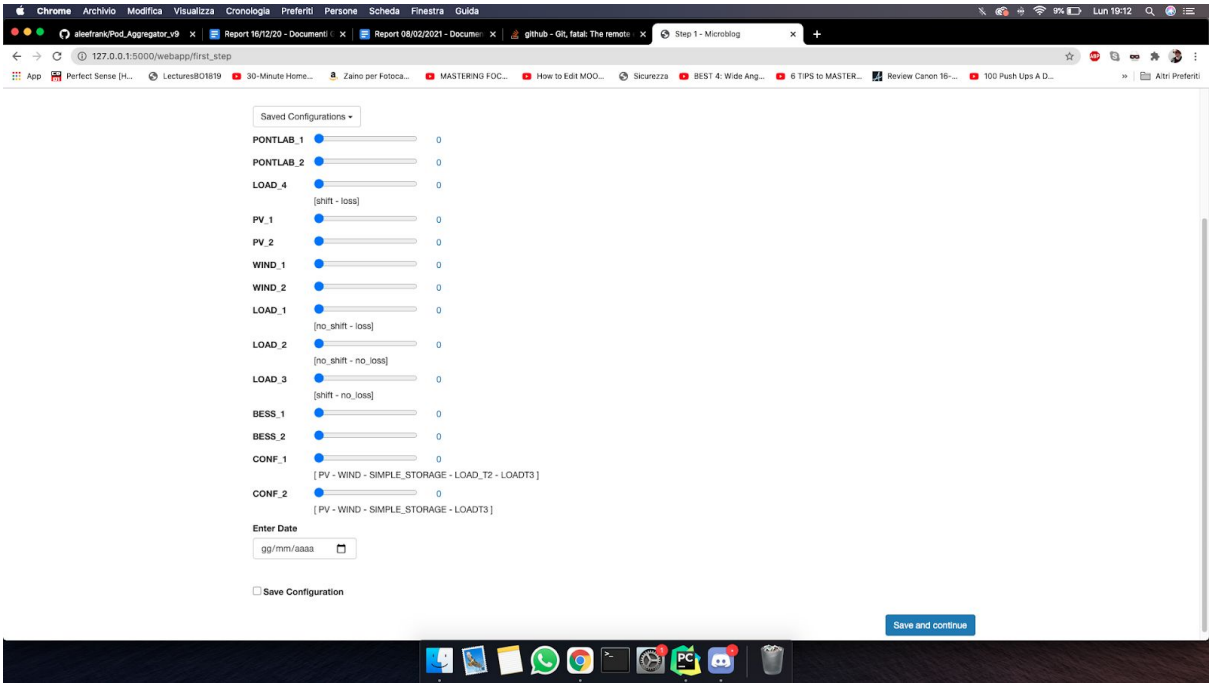
Qui viene anche effettuato il controllo in cui, per ogni istante di tempo, la flessibilità massima non debba essere inferiore alla minima. Qualora lo fosse, la massima assume il valore della minima.

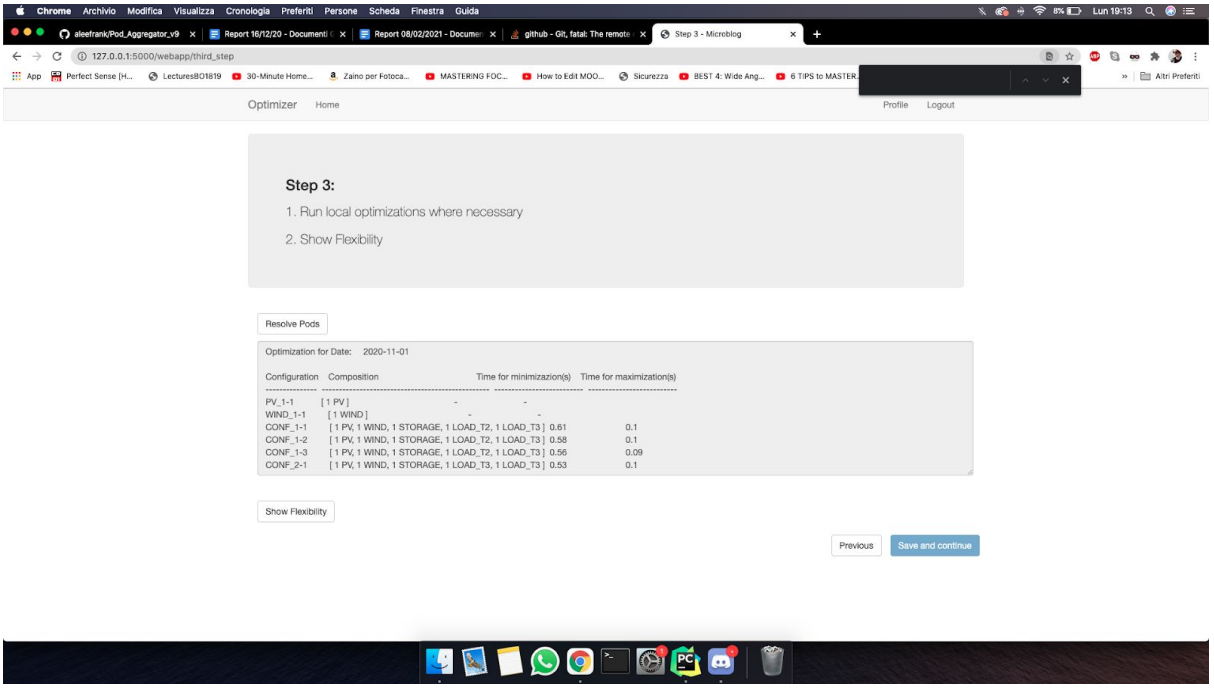
aggr_solver.py:
Solver per l'aggregatore.
Costruito con il dizionario *self.input*.
Crea un'istanza del modello, settando come parametri i dati presi nel costruttore

aggr_v2_model.py:
Seconda versione dell'aggregatore. (per capire se la logica implementata può avere senso preferirei fissare un ricevimento)

app.py:
Contiene le APIs per offrire il software come servizio web.
In mancanza di un collegamento diretto con Libra, è stato creato un dizionario per rappresentare la *uvax* nel modo più coerente possibile (vedere dizionario *uvax* in *app.py)*
Sono stati di conseguenza modificati gli algoritmi utilizzati dalle APIs descritti nel report precedente per recuperare Pods e relative baseline.
(N.B: attualmente le configurazioni miste CONF_* utilizzate dalle APIs non quelle finali. Sto testando le configurazioni finali nel file *aggr_main.py* sopra)

Optimizer-webapp-v1
Aggiunta indicazione dei pod che compongono le configurazioni miste [Fig.1]
Aggiunti risultati delle Ottimizzazioni Locali (minimizzazione, massimizzazione) [Fig.2]


Chrome Archivio Modifica Visualizza Cronologia Preferiti Persone Scheda Finestra Guida
Step 1 - Microblog
127.0.0.1:5000/webapp/first_step
Saved Configurations
PONTLAB_1 0
PONTLAB_2 0
LOAD_4 0
[shift - loss]
PV_1 0
PV_2 0
WIND_1 0
WIND_2 0
LOAD_1 0
[no_shift - loss]
LOAD_2 0
[no_shift - no_loss]
LOAD_3 0
[shift - no_loss]
BESS_1 0
BESS_2 0
CONF_1 0
[PV - WIND - SIMPLE_STORAGE - LOAD_T2 - LOADT3]
CONF_2 0
[PV - WIND - SIMPLE_STORAGE - LOADT3]
Enter Date
gg/mm/aaaa
Save Configuration
Save and continue


Chrome Archivio Modifica Visualizza Cronologia Preferiti Persone Scheda Finestra Guida
Step 3 - Microblog
127.0.0.1:5000/webapp/third_step
Optimizer Home
Profile Logout
Step 3:
1. Run local optimizations where necessary
2. Show Flexibility
Resolve Pods
Optimization for Date: 2020-11-01
Configuration Composition Time for minimization(s) Time for maximization(s)
PV_1-1 [1 PV] - -
WIND_1-1 [1 WIND] - -
CONF_1-1 [1 PV, 1 WIND, 1 STORAGE, 1 LOAD_T2, 1 LOAD_T3] 0.61 0.1
CONF_1-2 [1 PV, 1 WIND, 1 STORAGE, 1 LOAD_T2, 1 LOAD_T3] 0.58 0.1
CONF_1-3 [1 PV, 1 WIND, 1 STORAGE, 1 LOAD_T2, 1 LOAD_T3] 0.56 0.09
CONF_2-1 [1 PV, 1 WIND, 1 STORAGE, 1 LOAD_T3, 1 LOAD_T3] 0.53 0.1
Show Flexibility
Previous
Save and continue