Final Description of “IEEE-CIS Technical Challenge on Predict+Optimize for Renewable Energy Scheduling” Contribution

1st Steffen Limmer
Honda Research Institute Europe GmbH
63073 Offenbach am Main, Germany
steffen.limmer@honda-ri.de

2nd Nils Einecke
Honda Research Institute Europe GmbH
63073 Offenbach am Main, Germany
nils.einecke@honda-ri.de

***Abstract*—The present report describes our approach, which yielded the 3rd place in the final leaderboard of the IEEE-CIS Technical Challenge on Predict+Optimize for Renewable Energy Scheduling with a prediction error of 0.855737 and an energy cost score of 339160. For the load and PV predictions, we use simple straightforward approaches based on median values and machine learning, respectively. For the optimizations, we also use a standard approach, namely mixed integer linear programming, and applied different measures to improve its performance.**

I. Introduction

The task of the challenge is composed of three subtasks: Building load prediction, photovoltaics (PV) prediction, and the scheduling of activities and stationary batteries. Machine learning approaches and statistical models, like autoregressive integrated moving average (ARIMA), are common approaches for load forecasting. However, for the present problem, such approaches do not appear promising for two reasons: (1) The forecasting horizon is comparatively long, while having a fine-grained resolution of 15 minutes. (2) The amount of useful training data is limited since most of the historical data is pre-COVID. Thus, we decided to use a simple but robust approach for the load forecasting. PV prediction based on weather data is commonly done via machine learning approaches or physical models. We decided for the first, since it is easier to apply and does not require any additional information, like the orientation of the PV system. For optimizing the schedule of activities and batteries, we use a mixed integer linear programming[1] (MILP) approach. MILP is the predominant approach in the literature to energy management. It can efficiently handle constraints (the given problem has a lot of complex constraints) and it has the additional advantage of beeing able to provide guarantees on the optimality gap. However, since the given scheduling problem is very hard, even for a MILP approach, we applied different measures to accelerate the optimization.

[1]More precisely, a combination of mixed integer linear programming and mixed integer quadratic programming (MIQP) is used. For convenience, we do not differentiate between MILP and MIQP in the present manuscript.

II. Methodology

A. Building Load Prediction

We predict the building load in a certain time step of a week as median value over the load values in the corresponding time steps of the historical data, taking into account the data of eight weeks prior the month to predict. For example, the load at a Monday 10:00 am is predicted as the median over the loads at 10:00 am on Mondays in the historical data. We consider only eight weeks of historical data, since the characteristics of the load data is continuously changing since April 2020 due to restrictions in response to the COVID pandemic. Time steps for which no load data is available (”NaN” in the data) are not included in the computation of the median value and if all eight historical load values are not available, we predict a load of zero. We do not make use of weather data for the load prediction, since the impact of the weather on the load can be considered to be only marginal. We tried a few variations of the described approach, like using the mean instead of the median, or using other periods of historical data as inputs and decided for the final approach based on the prediction errors reported in the leaderboard of phase 1 of the challenge.

B. Photovoltaics Prediction

For the PV prediction, we use a machine learning approach with mainly weather data as input features. We cleaned the historical data for each of the six PV systems and removed days that contain (manually determined) outliers as well as days containing only zero-production entries. Furthermore, we shifted the weather data by -1 hour, since we realized a certain offset between solar radiation and PV production. In this way, the correlation between solar radiation and PV production in the training data is increased. For example, for the PV system Solar0, the Pearson correlation coefficient is 0.89 without the shift in the weather data and 0.92 with the shift. The preprocessed data is used to train a random forest with scikit-learn default hyperparameter setting. Since only hourly weather data is available and the prediction has to be done with a resolution of 15 minutes, the random forest predicts the PV production of a full hour in the form of four values (one per quarter of the hour). Let t denote a 15-minute time step corresponding to the start of a full hour. The random forest predicts the PV production values in time steps $t, \ldots, t+3$ based on the following inputs:

- Solar radiation in time step $t-4$
- Solar radiation in time step t
- Solar radiation in time step $t+4$
- Cloud coverage in time step $t-4$
- Cloud coverage in time step t
- Cloud coverage in time step $t+4$
- Thermal radiation in time step t
- Thermal radiation in time step $t+4$
- Ambient temperature in time step t
- Relative humidity in time step t
- Dewpoint temperature in time step t
- Atmospheric pressure in time step t
- Month corresponding to time step t
- Hour within the day corresponding to time step t

Thus, not only weather data from the hour to predict is used as input, but also data from the previous and the following hour. We came up with the described final approach by evaluating different approaches for the PV prediction for September 2020 and choosing the approach resulting in the smallest mean absolute errors. Besides random forest, we evaluated other models, like XGBoost, Gradient Boosting or kNN, we evaluated different hyperparameter settings for the random forest, and we also evaluated different input features.

C. *Optimization*

We formulated the optimization problem as a mixed integer linear programming problem and solved it with the help of the Gurobi[2] solver. A straightforward MILP formulation contains a large number of binary and integer variables, what typically has a negative impact on the solving time. For each activity, one binary variable is required for each possible start time of the activity. For each battery, two binary variables are required (one indicating charging and one indicating discharging) for each time step of the optimization horizon. For each combination of activity and building, multiple integer variables are required, which encode the number of rooms of the building, which are assigned to the activity. Furthermore, the objective function contains a quadratic term (square of the peak power), what makes the problem even harder to solve. Thus, in order to accelerate the optimization and to be able to compute solutions of high quality in an acceptable time, we applied different measures:

(1) Binary variables corresponding to start times of activities that make it impossible to satisfy the precedence constraints are excluded from the problem. For example, if a recurring activity cannot start on a Monday because it has a preceding activity, then for this activity no variables corresponding to start times on Monday are considered in the problem formulation. In this way, a significant reduction of the number of binary variables is achieved.

[2]https://www.gurobi.com/

(2) The assignment of buildings to activities is separated from the actual optimization. In the optimization, it is only ensured that a feasible assignment exists (meaning there is a sufficient number of rooms allowing to compute a feasible building assignment for the activity schedule at hand) without computing a concrete assignment. This notably reduces the number of integer variables and constraints considered in the optimization. In a post-processing step, the assignment of buildings to activities is computed by solving a second MILP problem, which only has to find a feasible assignment (the objective function is fixed to zero). This second problem can be solved within a few seconds.

(3) We used the parameter tuning tool of Gurobi in order to tune the parameters of Gurobi. In the final setting, we set the parameter `PreSparsify` to 0 and the parameter `VarBranch` to 1, while leaving the other parameters to their default values.

(4) The optimization is split into three subproblems, which are solved in sequence: First, the schedule of recurring activities is optimized assuming that the batteries are not used and that no once-off activities take place. Then, the battery charging is optimized under the assumption that the recurring activities take place according to the result of the first optimization. Then, the schedule of the once-off activities is optimized taking into account the results of the previous two optimizations. By decomposing the problem into easier subproblems, a notable acceleration can be achieved. Furthermore, it can be considered that the described problem decomposition has only a minor impact on the solution quality. Recurring activities can be scheduled only within office hours and it makes hardly sense to schedule additional once-off activities within office hours since this would increase the peak load. Thus, it should have no significant impact on the solution that the recurring activities and once-off activities are treated separately. In experiments, the order of the second two subproblems – first batteries and then once-off activities – yielded better results then the inverse order.

(5) In the optimization of the schedule of recurring activities, we do not use the original quadratic objective function. Instead, the peak load is linearly included in the objective function with a high coefficient of 1000. The scheduling of the recurring activities is the hardest of the three subproblems (at least for the given problem instances), and by linearizing its objective function it can be accelerated with only minor impact on the solution quality.

Our complete optimization consists of the execution of the following steps in the given order:

1) Schedule recurring activities (linear objective)
2) Schedule battery dis-/charging (quadratic objective)
3) Schedule once-off activities (quadratic objective)
4) Assign buildings to the scheduled activities (only search for a feasible solution)

III. Experiments

A. Predictions

In experiments, we evaluated our prediction on the October data from the first phase of the challenge. However, we removed the large outlier in the load of Building0 and filled the resulting gap via linear interpolation. Figure 1 shows the real and predicted total load together with the corresponding absolute prediction errors. Table I shows the mean absolute

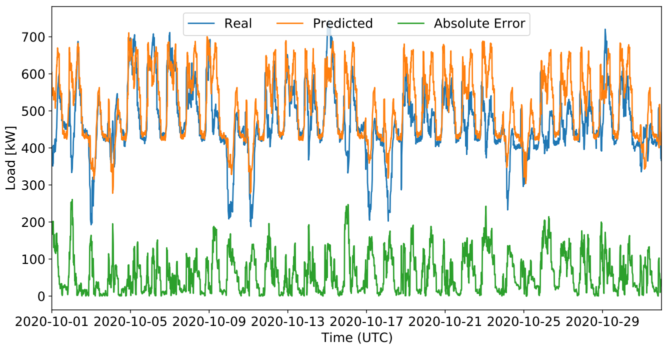

Fig. 1. Real and predicted October load and corresponding absolute error.

error (MAE) and root mean squared error (RMSE) for the different subloads and the total load. The PV prediction is

TABLE I
MEAN ABSOLUTE ERROR (MAE) AND ROOT MEAN SQUARED ERROR (RMSE) OF PREDICTIONS OF INDIVIDUAL LOADS/PRODUCTIONS AND TOTAL LOAD.

Load/Production	**MAE**	**RMSE**
Building0	21.26	41.39
Building1	1.11	1.75
Building3	31.05	42.74
Building4	0.78	1.01
Building5	8.17	15.13
Building6	2.60	3.65
Solar0	2.86	5.47
Solar1	0.71	1.42
Solar2	0.69	1.38
Solar3	0.65	1.26
Solar4	0.42	0.83
Solar5	2.19	4.33
Total Load	59.98	79.74

very accurate since the PV production is strongly correlated with the weather, which is used as input for the prediction. The building load prediction is more challenging – especially due to the impact of the COVID measures – but as one can see from Figure 1 the forecasted total load basically follows the real total load.

B. Optimizations

In order to investigate the effect of different improvements described in Section II-C, we executed optimizations according to the rules of phase 1 (meaning that the optimization period is October 2020 and the real load of the first 10 hours of the period is known) on six problem instances of phase 1. The prediction shown in Figure 1 is used as input for the optimizations. We ran the optimizations with six different settings:

- **c_q_f**: Combined optimization of all activities and batteries with quadratic objective and full number of binary variables.
- **c_q_r**: Combined optimization of all activities and batteries with quadratic objective and reduced number of binary variables (improvement (1)).
- **c_l_r**: Combined optimization of all activities and batteries with linear objective (improvement (5)) and reduced number of binary variables (improvement (1)).
- **s_q_f**: Separate optimizations of recurring activities, batteries, and once-off activities (improvement (4)) with quadratic objective and full number of binary variables.
- **s_q_r**: Separate optimizations of recurring activities, batteries, and once-off activities (improvement (4)) with quadratic objective and reduced number of binary variables (improvement (1)).
- **s_l_r**: Separate optimizations of recurring activities, batteries, and once-off activities (improvement (4)) with linear objective for recurring activities (improvement (5)) and reduced number of binary variables (improvement (1)).

Improvements (2) and (3) (separate building assignment and parameter tuning) are applied in all settings. We ran the experiments on a 3.8 GHz Intel Core i5-7600K quad-core CPU with 15.6 GB RAM and used version 9.1.0 of Gurobi as solver. For the combined optimizations, a time limit of 15 min is set and for the separate optimizations, time limits of 12 min, 1.5 min, and 1.5 min are set, respectively, for the optimizations of recurring activities, once-off activities and batteries. Table II shows the resulting electricity costs on the predicted load.

TABLE II
OPTIMIZATION RESULTS WITH DIFFERENT SETTINGS ON PROBLEM INSTANCES OF PHASE 1 WITH A TIME LIMIT OF 15 MIN FOR THE TOTAL OPTIMIZATION.

Problem	**c_q_f**	**c_q_r**	**c_l_r**	**s_q_f**	**s_q_r**	**s_l_r**
small0	47867	47026	43732	43715	43665	41571
small1	47662	46833	41044	41471	41444	39560
small2	49601	47695	41608	43598	42480	40333
large0	44699	44994	41831	42305	42009	41251
large1	43716	44400	41517	41513	41464	40872
large2	43474	43646	40245	41159	41171	40518
Mean	**46180**	**45766**	**41663**	**42294**	**42039**	**40684**

As one can see, the setting **s_l_r** (which corresponds to our final setting), yielded the best results on all problem instances. The benefit of the variable reduction (**c_q_r** vs. **c_q_f** and **s_q_r** vs. **s_q_f**) is small and in some cases – especially with the combined optimization – it has even a negative impact. However, as one can see, the linearization of the objective function (**c_l_r** vs. **c_q_r** and **s_l_r** vs. **s_q_r**) and the separate optimizations (**s_*** vs. **c_***) yield notable improvements.

Table III shows for problem instance small0 the cost on the predicted and real load (with removed outlier), when the optimization is done on the prediction (**pred-pred** and **pred-real**, respectively), and the cost on the real load, when the optimization is done on the real load (**real-real**). The optimizations were done with setting **s_l_r** and the previously stated time limits. The schedule resulting from the optimization on the prediction yields rather similar cost on the predicted and real load. By doing the optimization on the real load the cost on the real load is reduced by about 4.5%. This is mainly due to a reduction of the peak load. The optimization of the energy cost is not much influenced by the prediction error.

TABLE III
COST ON PREDICTED AND REAL LOAD RESULTING FROM OPTIMIZATIONS ON PREDICTED AND REAL LOAD ON PROBLEM INSTANCE SMALL0 OF PHASE1.

	pred-pred	**pred-real**	**real-real**
Peak cost	16246	18681	16896
Energy cost	25708	24341	24218
Once-off profit	383	383	383
Total cost	41571	42639	40731

C. Final Submission

Figure 2 shows the prediction in our final submission for phase 2. The prediction yielded a prediction error (mean absolute scaled error) of 0.855737 in the leaderboard. In phase 1, the same prediction approach yielded an error of 0.65888. We cannot tell, why the phase 2 error is notably higher than the phase 1 error.

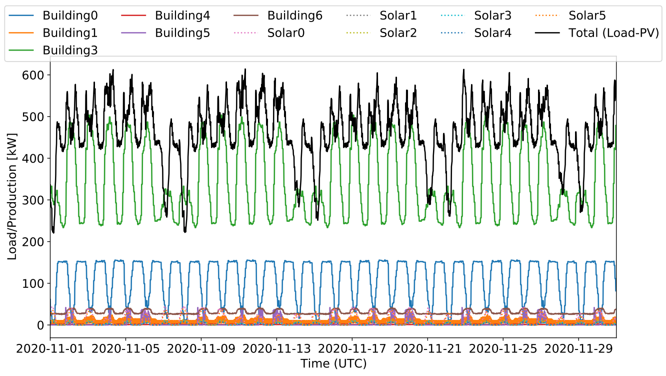

Fig. 2. Prediction of final submission phase 2.

We computed the optimization results of the final submission of phase 2 on a 2.1 GHz Intel Xeon Silver 4110 CPU with 8 cores and 93 GB RAM. We used version 9.1.0 of Gurobi as solver and set the time limits for the optimizations of recurring activities, batteries and once-off activities to 150 min, 15 min, and 15 min, respectively. Table IV shows the resulting optimality gaps for the three subproblems of the different problem instances as reported by the solver. For the instance large3, the solver did not find a lower bound in the optimization of once-off activities, and thus it could not report a gap. It can be seen that for four of the small instances, the subproblem of scheduling the recurring activities could be solved exactly. Not surprisingly, the optimizations of large instances yielded higher gaps than the optimizations of small instances. The optimization of once-off activities is the subproblem yielding the highest gaps. However, as one can see, the optimality gaps are generally small and one has to keep in mind that these are only worst-case gaps and the actual gaps might be smaller. On the predicted November load, the computed schedules yield total cost of A$ 234166, while on the real November load the total cost are A$ 339160 as reported in the challenge leaderboard. Thus, in contrast to the October data (see Table III), the difference is very large. We are not aware of the reason since we do not have access to the November data.

TABLE IV
PERCENTAGE OPTIMALITY GAPS ON PROBLEM INSTANCES OF PHASE 2 RESULTING FROM THE OPTIMIZATIONS USED TO COMPUTE THE FINAL SUBMISSION.

Problem	**recurring**	**batteries**	**once-off**
small0	2.5%	0.7%	2.7%
small1	0.0%	0.6%	2.5%
small2	0.0%	0.5%	3.2%
small3	0.0%	0.4%	1.9%
small4	0.0%	0.7%	0.0%
large0	1.4%	0.9%	5.3%
large1	1.4%	0.6%	4.7%
large2	1.5%	0.9%	4.5%
large3	1.6%	1.0%	-
large4	1.6%	0.9%	4.2%

Figure 3 exemplary shows the (predicted) load resulting from the schedule for problem instance small0 in our final submission. Like for several other problem instances in phase 2, no once-off activities are scheduled due to negative "profits" from scheduling them outside of office hours.

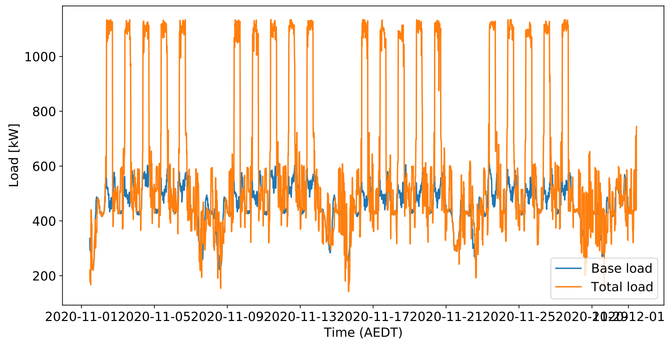

Fig. 3. Base load and total load resulting from schedule for instance small0 in final submission for phase 2.