Optimization heuristics for the residential energy load management

Claudio Giovanni Mattera

14th December 2012

In this thesis we dealt with the problem of optimizing the energy load management for a set of cooperating houses.

If we divide a day in many small intervals, and we measure how much energy is consumed by a house in each interval, we obtain the *energy demand curve*. In intervals where much energy is used the curve is high, and in those where little energy is used the curve is low. When considering multiple houses, the aggregate curve is the sum of single houses curves.

The more the demand curve is flat, regular and smooth, the more efficient the energy production and delivery is. If instead the curve is very irregular, with many isolated peaks, there are many disadvantages: power plants must continuously increase and decrease the production, and renewable energy sources are not suitable to supply energy for short peaks. The energy distribution grid must also be dimensioned to the highest peak, but most of the time much less energy is flowing.

Residential users tend to use their appliances with in a periodic fashion, all of them use the oven at lunch, the water boiler at breakfast... Each house has many isolated peaks, and different houses have peaks at the same time, which results in very irregular aggregate demand curves. In the medium and long term future all the electrical devices of a group of houses will instead be managed by a centralized system, which will schedule them to improve the shape of the demand curve.

We started from an initial model of this problem, which we briefly summarize in the following. We consider a set of houses, each with few electrical appliances. We divide a day in 15 minutes time slots; users set for each appliance a time window in which it can be executed. For instance, a user can set that the washing machine must start after 10:00 and must be finished before 19:00 (it does not set the exact starting time). In each time slot the active appliances consume energy, which is bought from the network. There is a limit on the energy absorbed in a single time slot, so it is not possible to schedule too many appliances at the same time. A centralized system takes care of collecting all these data and to compute the starting time of each appliance. The goal is to optimize a given function of the aggregate demand curve, for instance to minimize its maximal height.

We also consider two extensions: a house can have photovoltaic panels and/or batteries. The former produce energy in a unmanageable way, the latter can be used to store energy for later usage.

There exists a mathematical programming formulation for this model, that can be used to produce the optimal scheduling, but this requires extremely long computing time for all but the smallest instances (an *instance* is a particular set of problem's data, large instances have many houses and many appliances, which means many variables). The purpose of our thesis was to develop an heuristic, that is, a method to generate high quality solutions in short time. A *solution* is a synonym for a scheduling for all the appliances (plus the information on batteries usage), and its quality depends on a given function of the resulting demand curve (for instance, a scheduling which produces a regular curve has high quality).

In order to generate an initial solution we used

an algorithm called GRASP. We start from an empty scheduling, which corresponds to a flat demand curve, then we add each appliance for each house one after another. For each appliance, we try to schedule it in all the possible starting time slots in the time window, computing the resulting demand curve. We discard the worst alternatives and finally we take the actual starting time slot randomly among the remaining ones. When we scheduled all appliances we obtained an initial solution. We repeat the whole procedure many times, and in the end we keep the best solution generated.

In Figure 1 we show an example of adding an appliance. We try to schedule the new appliance ▬ in all time slot, and we show the resulting demand curves. Starting slots 4 and 5 results in the lowest curves.

GRASP is extremely fast to generate initial solutions, but their quality is rather low. In order to improve an existing solution we used a local search iterative algorithm, that is, from a solution k we produce a slightly better solution $k+1$, from which we produce a slightly better solution $k+2$ and so on... Solution's quality improves at each iteration.

Starting from a solution k we generate a neighbourhood, that is, a set of solutions close to k. For instance we can consider all the scheduling that can be obtained from the current one by changing the starting time slot of an appliance. We then take the best neighbour, according to the resulting demand curve, as the next solution. If we stop whenever we can not find in the neighbourhood a better solution than the current one, the algorithm is called *steepest descent method*. Unfortunately it can happen that, even if we can not find a better solution, the current one is suboptimal. We say that we stopped in a *local* optimum, because at local level it seems optimal, but it is not. For instance, if we minimize the altitude in a region, the bottom of a mountain hollow is locally the lowest point, but the global optimum is probably on the seashore. Tabu Search is a method based on local search that mitigates this problem.

In Tabu Search we remove from the neighbourhood the solutions we already visited, and we allow non-improving iterations. Whenever we reach the bottom of a local hollow, we continue climbing up its walls; we can not fall inside again, since solutions inside the hollow are forbidden.

This approach produced good quality solutions in very short time if there were no batteries.

In order to use batteries along local search we define another type of neighbourhood. We consider all the scheduling that can be obtained from the current one by: identifying a time slot t_d where some energy is bought, buying that amount of energy in an earlier time slot t_c, storing it in a battery, releasing it in t_d. We literally shift peaks into valleys in the demand curve.

This approach produced medium quality solutions in very short time.

Another way to improve a solution is to obtain a *reduced problem* from it, that is, a smaller instance with less variables and parameters, and solving it using the mathematical programming formulation. For each appliance we randomly reduce the time window in which it can start: we keep the current starting time slot and about 40% of the others. Using the mathematical programming formulation we could solve small instances in reasonable time.

This method produced high quality solutions even when there were batteries, but in slightly longer time. Moreover, such solutions could be further refined with Tabu Search.

In Figure 2 we show an example of solutions computed with the methods we developed by minimizing the height of the demand curve, for the two cases where batteries are available or not. The horizontal line shows the average height, the best curve we could hope for (although it is unlikely that exists a scheduling with such curve).

The low quality curve have many peaks and it is quite high. High quality curves are instead much more regular and lower. Ignoring batteries, we have a valley at earlier hours at night, since only few appliances can be scheduled and little energy is bought. If we use batteries, on the other hand, we can charge them in those hours, in order to decrease the curve later in the day. Both high quality curves are within 102% of the optimal one.

Figure 1: Adding an appliance with GRASP

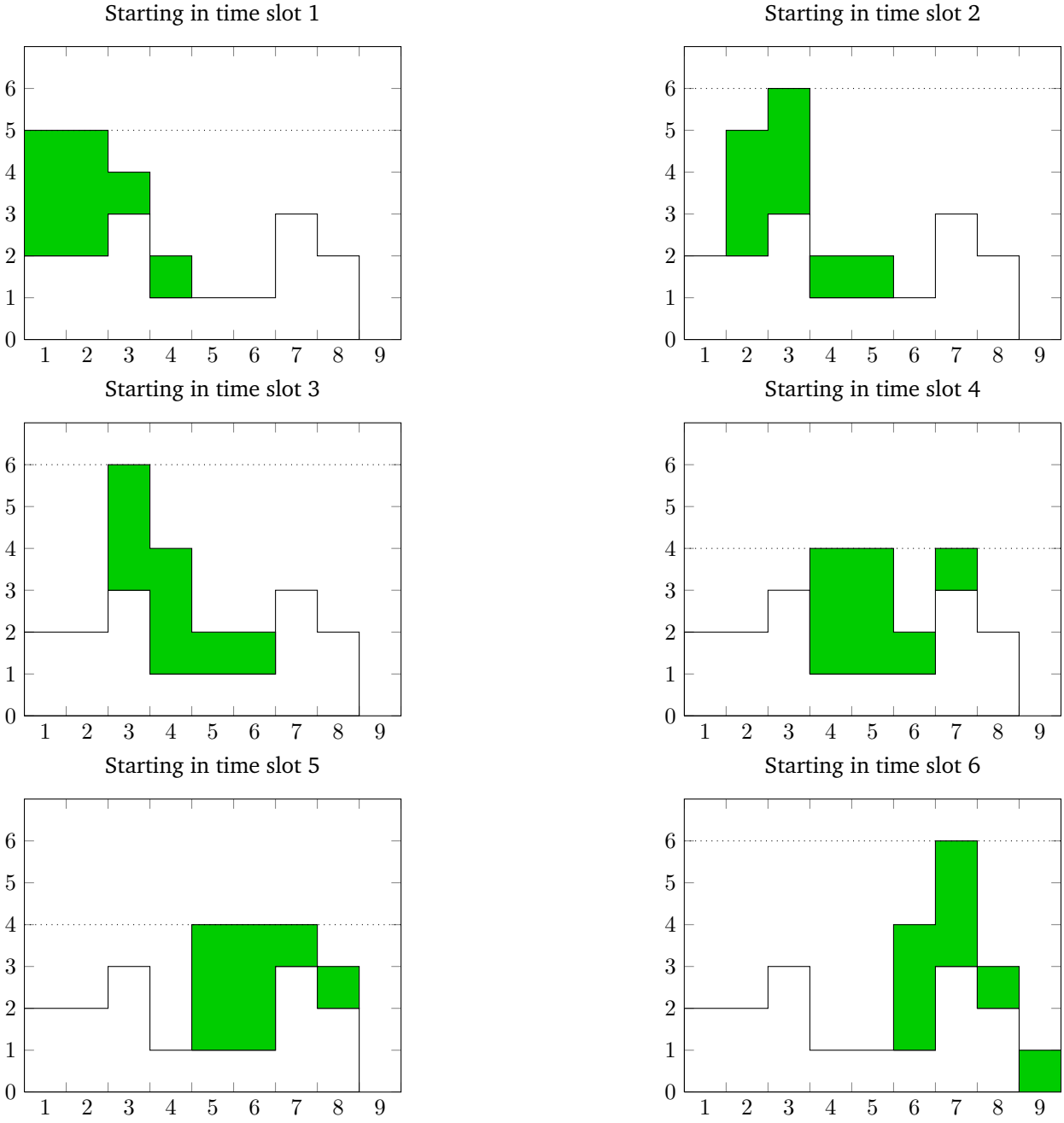
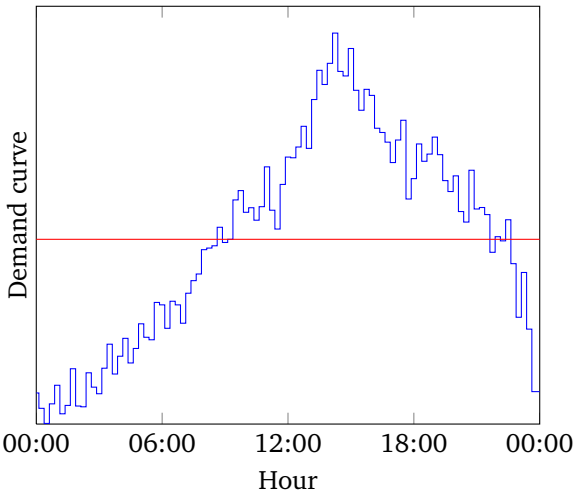
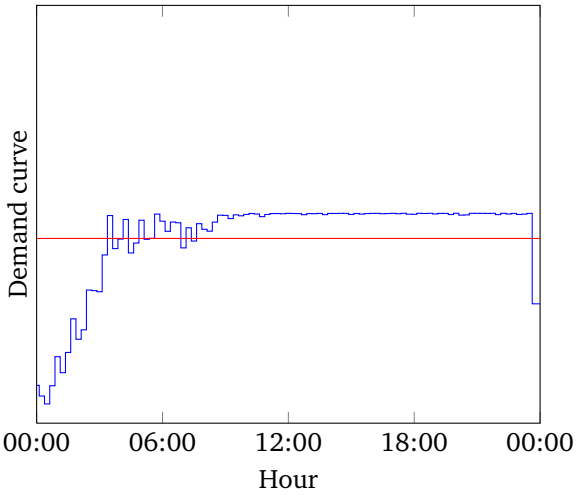


Figure 2: Solutions

Low quality solution



High quality solution, ignoring batteries



High quality solution, using batteries