A MIP-and-refine matheuristic for smart grid energy management

Matteo Fischetti*[1], Giorgio Sartor†[1], and Arrigo Zanette‡[1]

[1]DEI, University of Padova

October 2012; Revised, 13 March 2013

Abstract

In the last years we have witnessed an increasing interest in smart buildings, in particular for what concerns optimal energy management, renewable energy sources, and smart appliances. In this paper we investigate the problem of scheduling smart appliance operation in a given time horizon with a set of energy sources and accumulators. Appliance operation is modeled in terms of uninterruptible sequential phases with a given power demand, with the goal of minimizing the energy bill fulfilling duration, energy, and user preference constraints. A Mixed Integer Linear Programming (MIP) model and a greedy heuristic algorithm are given, intended to be used in a synergic way. We show how a general purpose (off-the-shelf) MIP refining procedure can effectively be used for improving, in short computing time, the quality of the solutions provided by the initial greedy heuristic. Computational results confirm the viability of the overall approach, in terms of both solution quality and speed.

Keywords Matheuristics, Mixed-Integer Programming, Refinement heuristics, Energy Management, Smart Houses

*matteo.fischetti@unipd.it

†gio.srt@gmail.com

‡zanettea@gmail.com

1 Prologue

Many successful matheuristic schemes use a black-box MIP solver to generate high-quality heuristic solutions for difficult optimization problems. The hallmark of this approach is the availability of a (possibly incomplete) MIP model of the problem at hand, and of an external metascheme that iteratively solves sub-MIPs obtained by introducing invalid constraints (e.g., variable fixings) defining "interesting" neighborhoods of certain solutions. The goal of the approach is to iteratively refine the incumbent solution, producing a sequence of better and better feasible solutions in short (or, at least, acceptable) computing times. The above solution-refinement approach is completely general, i.e., it can in principle be applied to the original MIP without the need of ad-hoc adaptations.

An example of a general MIP refinement procedure is the evolutionary *polishing method* of Rothberg [7]. The method implements an evolutionary metaheuristic to be applied at selected nodes of a branch-and-bound tree. A fixed-size population of feasible solutions is maintained. Iteratively, two or more "parent" solutions are combined with the aim of creating a new "son" member of the population. This is done by fixing all variables whose value coincides in the parents solutions, and by heuristically solving the resulting sub-MIP by invoking an external MIP solver for a limited number of branch-and-bound nodes. Diversification is guaranteed by performing a classical mutation operation that consists in (i) selecting at random a seed solution in the population, (ii) fixing at random some of its variables, and (iii) heuristically solve the resulting sub-MIP.

Interesting enough for practitioners, an off-the-shelf implementation of the polishing heuristic is available in some commercial MIP solvers, hence it can readily be used.

In very difficult cases, however, the approach based on general MIP refinement is not successful, and one tends to design ad-hoc matheuristics that exploit the structure of the problem. As a matter of fact, a main issue with the general approach is the lack of good (or even feasible) solutions to refine. In this context, one can argue that ad-hoc heuristics and general

MIP refinement procedures are complementary one to each other, the former being typically able to find feasible solutions very quickly, while the latter can exploit the underlying MIP model to improve them by reaching a quality degree that is difficult to attain otherwise.

In the present paper we show how the application of above scheme can lead to a very effective overall heuristic even in case a very simple greedy is used to feed an off-the-shelf general MIP refinement module. The resulting *MIP-and-refine* approach is exemplified and tested in the context of smart grid energy management, whose underlying MIP models turn out to be very difficult to solve without the hints provided by an external heuristic.

Although we cannot claim deep theoretical contributions, we hope that the present paper can be used as a case study by researchers and practitioners working in the field of matheuristics, and that the simplicity of the MIP-and-refine approach will make it one of the first options to try when approaching a new problem.

2 Introduction

Energy optimization is attracting an increasing interest amongst researchers as long as new "smarter" infrastructures and devices are going to replace the traditional ones. The most popular scenario involves a new concept of electrical grid, the *Smart Grid* that allows to convey a two-way flow of electricity and information between central generators and customers [5]. Smart Grid benefits are fully exploited only if the grid endpoints, home appliances for examples, are smart as well. Smart appliances are able to exchange data with the grid, such as dynamic energy prices and grid status. Along with user preferences, this information can be used to optimally manage the energy demand in order to reduce the customer energy bill and to prevent major blackouts. A common way of addressing the demand side energy management problem is by solving a scheduling problem involving multiple appliances with different operational constraints, user preferences, renewable energy sources, and batteries.

Mixed Integer Linear Programming (MIP) models from the literature

allow for an effective mathematical formulation of the appliance scheduling problem. Barbato et al. [2] also take the photovoltaic energy into account, and a linearized description of battery charge states is given. Sou et al. [9] provide a detailed MIP formulation of appliance power profiles and operations, and model appliance operations as a set of sequential uninterruptible phases with variable inter-phase delays. Other authors have investigated variants of the appliance scheduling problem, Hatami and Pedram [6] by taking the interaction among different users into account, Zhang et al. [10] by considering a so-called microgrid, and Agnetis et al. [1] by addressing additional thermal comfort constraints.

As far as the solution of appliance scheduling problem is concerned, Carpentieri et al. [4] propose an LP-rounding heuristic for solving the appliance scheduling problem with the goal of minimizing the maximum peak energy of multiple houses. Barbato et al. [3] use different heuristics to address the problem of online recovering an offline schedule taking into account the real parameters.

The paper is organized as follows. In Sections 3 and 4 we describe the only two ingredients of our recipe: the MIP model and a simple greedy heuristic, respectively. Computational results are then presented in Section 5, while some conclusions are drawn in Section 6.

The present work is based on the dissertation of the second author [8].

3 Our MIP model

Following Sou et al. [9], we model appliance operations as a set of sequential uninterruptible energy phases, each of which uses a given total amount of electric energy. For example, typical washing machine phases are pre-wash, wash, rinse and spinning. The set of phases is also called the *power profile* of the appliance.

Depending on the appliance, phase duration may vary, as long as the inter-phase delay (e.g., the spinning of the washing machine must start within ten minutes of the rinsing being finished). The total energy given for a phase can be evenly distributed over time, or it may vary. We model the

latter case with per-phase bounds on the instantaneous power consumption.

Besides the intrinsic operational constraints, we allow the user to specify preferences for the time interval in which an appliance should run (e.g., start the washing machine between 4pm and 6pm).

Following Barbato et al. [2], we have modeled three classes of energy sources: power grid, domestic renewable energy and accumulators (batteries). The power grid advertises the maximum amount of available energy (*peak power*) for each time instant. Note that this peak power can be different from the actual user contract maximum power. In fact, a common feature in the Smart Grid paradigm is to dynamically advertise (i.e., broadcast) the peak power depending on the grid state, in order to let users adjust their demands for preventing more dangerous power outages. Along with the peak power, also the cost of energy changes with time. For example, in the Italian market it can vary between two values depending on the day time and on the day of the week. More dynamic power grids allow for a finer grain price adjustment (hourly or less).

Domestic renewable energy sources provide free energy but with a limited availability. For example, the performance of a photovoltaic (PV) plant depends on geographical position, weather conditions, and time. Accumulators allow to store energy (from grid or from other sources) when energy price is low, and to use it later when energy price is higher. This feature represents an important degree of freedom as far as optimization is concerned. Our model only deals with batteries, viewed as direct electric energy accumulators; however, it can trivially be extended to other types of energy accumulators (e.g., boilers for thermic energy).

Finally, the optimization goal is to minimize the total energy cost by finding a proper allocation of all appliance phases.

Given two integers a and b, let $[a, b]$ denote the discrete set $\{a, a + 1, \cdots, b\}$. We discretize the scheduling time horizon into m uniform time slots, indexed by $k \in [1, m]$. The phases for each appliance $i \in [1, N]$ are denoted by $j \in [1, n_i]$. To simplify notation, in what follows we write $\forall i, j$ instead of $\forall i \in [1, N], j \in [1, n_i]$, and $\forall k$ instead of $\forall k \in [1, m]$.

In our model, nonnegative continuous variables p_{ij}^k represent the energy

assigned to phase j of appliance i during time slot k. The typical unit for p_{ij}^k is Watt (W) per timeslot (energy). With binary variables x_{ij}^k, s_{ij}^k and t_{ij}^k we model the allocation of a time slot k for phase j of appliance i. In particular, $x_{ij}^k = 1$ iff phase j of appliance i is allocated in time slot k. Variable s_{ij}^k jumps from 0 to 1 right after the last time slot of where the phase j of appliance i is allocated, and is defined by the equations:

$$x_{ij}^{k-1} - x_{ij}^k \leq s_{ij}^k \qquad \forall\ i, j, \forall k \in [2, m] \tag{1a}$$

$$s_{ij}^{k-1} \leq s_{ij}^k \qquad \forall\ i, j, \forall k \in [2, m] \tag{1b}$$

Instead, t_{ij}^k is 1 iff there is a inter-phase delay between phase $j-1$ and j in time slot k, and is defined as:

$$t_{ij}^k = s_{i(j-1)}^k - (x_{ij}^k + s_{ij}^k) \qquad \forall i, k, \forall j \in [2, n_i]$$

Figure 1 illustrates the meaning of the above variables in a simple case of an appliance with two phases: the first phase is allocated between hours 2 and 6, and the second between 14 and 16 (the day being divided into 12 time slots).

Our model also uses nonnegative continuous variables z^k and y^k to represent the amount of total energy sold and bought in each time slot k, respectively. Then, if c^k and g^k denote the input cost of bought and sold electricity during time slot k, respective ly, our MIP model calls for

$$z = \min \sum_{k=1}^{m} \left(c^k y^k - g^k z^k \right) \tag{2}$$

subject to the following constraints.

Phase energy: ensures the energy allocated during phase j of appliance i meets the given phase total energy E_{ij}

$$\sum_{k=1}^{m} p_{ij}^k = E_{ij} \qquad \forall\ i, j \tag{3}$$

Energy bounds: ensures the energy allocated in phase j for appliance i in

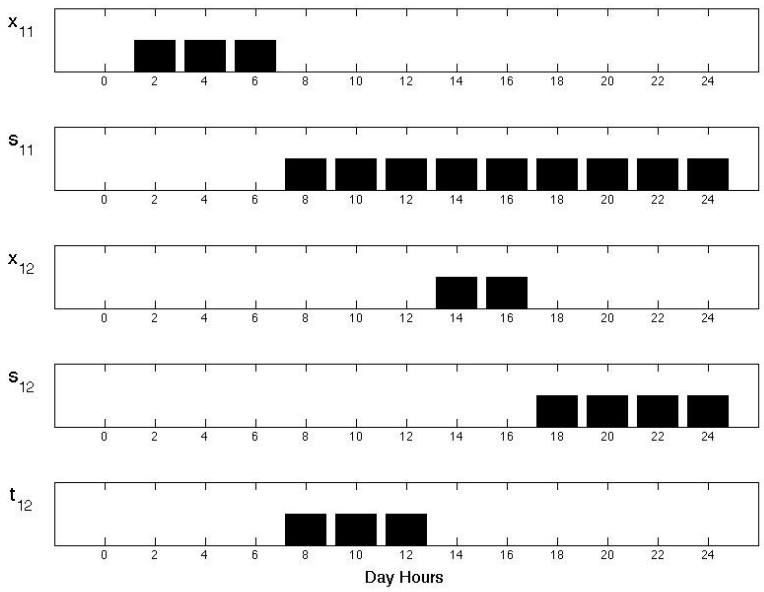

Figure 1: Example of binary variables x_{ij}^k, s_{ij}^k and t_{ij}^k

any time slot k belongs to the allowed range $[\underline{P}_{ij}, \overline{P}_{ij}]$

$$\underline{P}_{ij}\ x_{ij}^k \leq p_{ij}^k \leq \overline{P}_{ij}\ x_{ij}^k \qquad \forall\ i, j, \forall k \tag{4}$$

Power safety: guarantees that the total energy assigned in time slot k does not exceed the peak power limit

$$y^k \leq P_{peak}^k \qquad \forall\ k \tag{5}$$

where P_{peak}^k is the peak limit of slot k; this constraint can also be used to model additional unscheduled power demands that reduce the available grid energy in time slot k.

Energy phase duration:

$$\underline{T}_{ij} \leq \sum_{k=1}^{m} x_{ij}^k \leq \overline{T}_{ij} \qquad \forall\ i, j \tag{6}$$

where $\underline{T}_{ij}$ and $\overline{T}_{ij}$ represent, respectively, the lower and upper bound on the number of time slots to allocate for phase j of appliance i.

Uninterruptible phase: these constraints ensure that all time slots of phase j are allocated contiguously (i.e., when an energy phase starts, it must finish without interruptions).

$$x_{ij}^k + s_{ij}^k \leq 1 \qquad \forall\ i, j, k \tag{7}$$

Recall that s_{ij}^k is 0 before the last time slot allocated for phase j and appliance i, and becomes 1 afterwards (1a) until the end (1b). Thus, constraint (7) prevents the variable x_{ij}^k to be 1 after the last-phase time slot.

Sequential processing: previous energy phase must be finished, before a new one starts

$$x_{ij}^k \leq s_{i(j-1)}^k \qquad \forall\ i, k,\ \forall\ j \in [2, n_i] \tag{8}$$

Inter-phase delay duration:

$$\underline{D}_{ij} \leq \sum_{k=1}^{m} t_{ij}^{k} \leq \overline{D}_{ij} \qquad \forall\ i,\ \forall\ j \in [2, n_i] \tag{9}$$

where $\underline{D}_{ij}$ and $\overline{D}_{ij}$ are the minimum and the maximum number of time slots between phase $j-1$ and j of the appliance i.

User time preferences: disable phase allocation of appliance i in the given time slots

$$x_{ij}^{k} \leq TP_{i}^{k} \qquad \forall\ i, j, k \tag{10}$$

where TP_{i}^{k} is equal to zero iff phase j of appliance i cannot be allocated in time slot k.

In order to model batteries behavior we need two extra binary variables w_c^k and w_d^k, where w_c^k is equal to 1 if the battery is charging in time slot k and 0 otherwise, while w_d^k is equal to 1 if the battery is discharging in time slot k and 0 otherwise. Moreover, with the nonnegative continuous variables v_c^k and v_d^k we describe the charge and discharge rates, respectively, that is the amount of energy that is charged/discharged in time slot k. The total accumulated energy in time slot k is described by the nonnegative continuous variable e^k.

Battery usage constraint: the battery cannot charge and discharge at the same time.

$$w_c^k + w_d^k \leq 1 \qquad \forall\ k \tag{11}$$

Battery charge/discharge rate bounds:

$$v_c^k \ \leq \ v_c^{max} \cdot w_c^k \qquad \forall\ k \tag{12a}$$

$$v_d^k \ \leq \ v_d^{max} \cdot w_d^k \qquad \forall\ k \tag{12b}$$

where v_c^{max}, v_d^{max} denote the max charge and discharge rates respectively.

Battery energy function: this is a linearization of the actual charge/discharge curves

$$e^k = e^{k-1} + \eta_c \cdot v_c^k - \eta_d \cdot v_d^k \qquad \forall\ k \tag{13}$$

where η_c and η_d are, respectively, the charging and discharging efficiency

Battery capacity bounds: used to limit the energy stored in the battery

$$\gamma^{min} \leq\ e^k\ \leq\ \gamma^{max} \qquad\qquad \forall\ k \tag{14}$$

where γ^{max} and γ^{min} represent the maximum capacity and the minimum energy safety value (required, for example, by lithium batteries)

Balancing constraint: balance between produced and consumed energy

$$y^k + \pi^k + v_d^k = z^k + \sum_{i=1}^{N} \sum_{j=1}^{n_i} p_{ij}^k + v_c^k \qquad\qquad \forall\ k \tag{15}$$

where π^k is the the sum of the newable domestic power sources contribution in time slot k.

4 A Greedy Algorithm

In this section we describe a heuristic greedy algorithm for finding good feasible solutions of the described problem, that we apply in a multi-start fashion. The algorithm schedules appliances in order of decreasing priority, according to a greedy policy—once an appliance has been scheduled, it is not changed anymore, and all other appliances are allocated on top of the current partial solution. In the first application of the greedy, we use energy requirements as appliance priorities. In the subsequent runs, the priority vector is shifted to generate different solutions. For each appliance, we look for a feasible allocation of its phases according to the following rules.

We consider a simplified (more restricted) version of the problem, where the duration d_{ij} (say) of each phase is the minimum between $\overline{T}_{ij}$ and $\lceil E_{ij}/\underline{P}_{ij} \rceil$, and bounds (4) become $p_{ij}^k = x_{ij}^k E_{ij}/d_{ij}$ for all k.

Accordingly, every phase has a constant duration and a constant energy consumption, and can be scheduled in the time slots interval $[1, m - d_i^{min}]$ where $d_i^{min} = \sum_{j=1}^{n_i} d_{ij}$ represents the minimum duration of a complete appliance power profile (i.e., without phase delays). To be more specific, once a phase has been allocated we look for all possible allocations of the next phase in the range given by $[\underline{D}_{ij}, \overline{D}_{ij}]$, see (9), and we select the most profitable one. Our allocation procedure enforces the user preferences on time slots (10) and three other constraints: power safety (5), uninterruptible phase (7), and sequential processing (8).

A pseudo-code of our heuristic is given in Algorithm 1. We have an external loop (starting after line 1) where appliances i are scheduled one after the other. For each appliance i, we consider all possible "shifts" t for the starting slot of the first phase (line 3), and then consider a straightforward greedy (lines 8-19) to find the best starting slot $k_j^\star$ of all other phases j. Variable k_{min} gives the first slot k currently available for allocation: it is initialized at line 7 after the allocation of the first phase, then it is updated at line 18 after the allocation of each new phase.

Recall that we consider a simplified phase allocation whose duration d_{ij} is a constant, so the "cost of allocating i, j starting from slot k" at lines 5 and 11 refers to the solution with $x_{ij}^h = 1$ for all $h \in [k, k + d_{ij} - 1]$, and also takes user preferences into account.

When all possible shifts t have been tried, $z^\star$ gives the cost of the (heuristically) best assignment for all phases of i, the corresponding solution being stored in the incumbent x. At line 27, if no feasible phase allocation was found for appliance i (case $z^\star = +\infty$) the overall problem is heuristically considered to be infeasible and the greedy is aborted—although this situation can occur if the constraints are very tight, it never occurred in our tests.

As to time complexity, the most time consuming step is the cost evaluation at line 11, that takes $O(d_{max})$ time, where $d_{max} = \max_{ij} d_{ij}$. This step is executed, for each appliance i and for each shift t, at most $n_{max} D_{max}$ times, where $n_{max} = \max_i n_i$ and $D_{max} = \max_{ij}\{\overline{D}_{ij} - \underline{D}_{ij} + 1 : j \geq 2\}$. As we have N appliances i and $O(m)$ possible shifts t, the overall time

complexity of our heuristic is $O(Nn_{max}D_{max}d_{max}m)$.

5 Computational results

In our tests we considered a time horizon of 24 hours, subdivided in 96 time slots of 15 minutes each. Experiments were grouped into four sets according to two main parameters. The first parameter is the "flexibility" of the *user time preference* constraint. We considered two level of flexibility, namely: *high flexibility* (HF), meaning that the appliance can be scheduled at any time during the day, and *medium flexibility* (MF), meaning that appliances can run inside a 12-hour randomly-generated time window within the day. The second parameter is electricity cost: it can vary either every two hours (BC), or every time slot (TC). For each of the four resulting sets, namely HFBC, HFTC, MFBC, and MFTC, we considered 10, 20, or 30 appliances, respectively, and solved 5 random instances for each of the 12 combinations—60 instances in total.

A constant price of the sold photovoltaic energy was considered, equal to half of the minimum cost of the bought energy. All the other model parameters are taken from uniform random distributions: $c_k \in [2, 4]$, $j \in [2, 5]$, $E_{ij} \in [400, 800]$, $\underline{P}_{ij} \in [50, 80]$, $\overline{P}_{ij} \in [400, 800]$, $\underline{T}_{ij} \in [1, 2]$, $\overline{T}_{ij} \in [3, 5]$, and $P^k_{peak} \in [2400, 2600]$. For all appliances i, we set $\underline{D}_{ij} = 0$ for all j, and $\overline{D}_{ij} \in [4, 6]$ (if $j \geq 2$) or $\overline{D}_{ij} = m - \sum_t d_{it}$ (if $j = 1$).

We considered a single renewable photovoltaic power source, whose provided energy is sampled from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with mean $\mu = 52$ (1pm, the period of maximum production at the latitude of Italy), standard deviation $\sigma = 10$, and maximum value of 1250W per time slot. The considered battery has a capacity $\gamma^{max} = 500$ Watt per time slot and charge/discharge rates $v_c^{max} = v_d^{max} = 50$ Watt per time slot, with efficiencies $\eta_c = \eta_d = 1$.

Algorithm 1 Greedy algorithm

for all appliances $i \in [1, N]$ by nonincreasing $\sum_j E_{ij}$ **do**
  $z^\star \leftarrow +\infty$; ▷ Best allocation cost for appliance i
  for all $t \in [1 + \underline{D}_{i1}, 1 + \overline{D}_{i1}]$ **do** ▷ Try all possible shifts t
    $j \leftarrow 1$;
    $z \leftarrow$ cost of allocating i, j starting from slot t ($+\infty$ if infeas.);
    $k^\star_j \leftarrow t$;
    $k_{min} \leftarrow t + d_{ij}$; ▷ First slot k available for the next phase of i
    for all phases $j \in [2, n_i]$ **do** ▷ Greedy alloc. of all other phases
      $\overline{c} \leftarrow +\infty$; $\overline{k} \leftarrow -1$;
      for all $k \in [k_{min} + \underline{D}_{ij}, k_{min} + \overline{D}_{ij}]$ with $k + d_{ij} - 1 \leq m$ **do**
        $c \leftarrow$ cost of alloc. i, j starting from slot k ($+\infty$ if infeas.);
        if $c < \overline{c}$ **then**
          $\overline{c} \leftarrow c$; $\overline{k} \leftarrow k$
        end if
      end for
      $z \leftarrow z + \overline{c}$;
      $k^\star_j \leftarrow \overline{k}$;
      $k_{min} \leftarrow \overline{k} + d_{ij}$
    end for
    if $z < z^\star$ **then** ▷ Update the best allocation for appliance i
      $z^\star \leftarrow z$;
      $x^k_{ij} \leftarrow 0$ for all j, k ;
      $x^k_{ij} \leftarrow 1$ for all j and $k \in [k^\star_j, k^\star_j + d_{ij} - 1]$
    end if
  end for
  if $z^\star = +\infty$ **then** ▷ No feasible allocation found for appliance i
    return infeasible
  end if
end for
return the incumbent solution x

We compared four different solution approaches, all run in single-thread mode:

- `Greedy-alone`: our stand alone greedy algorithm without multistart enhancement;
- `Greedy`: our greedy algorithm applied N times by starting from the N possible shifts of the initial priority vector, taking the best solution found and storing the others;
- `Cplex`: the state of the art IBM ILOG CPLEX MIP 12.4 solver used as a black-box, with its default setting, stopped as soon as the first feasible solution is found;
- `Cplex+Polish`: CPLEX's polishing refining heuristic [7] applied right after the root node and for a total of 10 nodes (all cuts disabled), when starting from the feasible solution found by the previous `Cplex` algorithm;
- `Greedy+Polish`: our proposed MIP-and-refine scheme, i.e., the previous `Cplex+Polish` algorithm but starting from the list of all feasible solutions found by `Greedy`.

Note that the last three methods also return a lower bound on the optimal value hence, as a byproduct, they can provide a proof of optimality in some (easy) cases. This is true, in particular, when the root-node lower bound is very tight and the method starts with an (almost) optimal `Greedy` solution, meaning that just few branching nodes need to be explored.

	Greedy			Cplex			Cplex+Polish			Greedy+Polish		
Set	% Incr	% STD	Time	% Incr	% STD	Time	% Incr	% STD	Time	% Incr	% STD	Time
HFBC 10	0.0 ^{5}	0.0	0.5	252.9	193.4	56.8	72.7 ^{2}	95.6	94.9	0.0 ^{5}	0.0	0.6
HFBC 20	-0.5	0.5	2.4	50.2	80.4	3,054.4	13.4	41.2	3,223.2	-9.4	4.8	262.0
HFBC 30	-0.6	0.5	6.4	1.8	4.7	9,585.8	-5.1	3.4	9,966.5	-6.1 ^{1}	3.2	310.2
HFTC 10	0.0 ^{5}	0.0	0.4	94.0	129.2	49.0	47.0 ^{4}	94.1	83.4	0.0 ^{5}	0.0	0.6
HFTC 20	-1.7	1.2	2.3	58.0	49.9	2,213.0	-1.4	21.3	2,387.4	-22.2 ^{1}	1.8	154.7
HFTC 30	-0.6	0.3	6.6	-12.3	1.4	46,439.1	-19.0	2.6	46,904.7	-21.2 ^{1}	2.5	617.7
MFBC 10	0.0 ^{5}	0.0	0.3	157.3 ^{1}	146.7	5.2	38.8 ^{2}	51.4	15.4	0.0 ^{5}	0.0	1.3
MFBC 20	-3.0	2.2	1.5	160.1	45.9	35.2	24.4	24.2	90.0	-13.8	7.8	100.8
MFBC 30	-0.8	0.5	3.7	44.4	11.5	65.1	0.7	8.3	164.5	-11.7 ^{1}	4.5	67.4
MFTC 10	0.0 ^{5}	0.0	0.3	159.9	89.5	5.6	33.5 ^{1}	24.8	18.8	0.0 ^{5}	0.0	1.5
MFTC 20	-1.2	1.2	1.5	54.8	21.8	52.5	-8.4	13.6	114.2	-23.5 ^{1}	2.6	54.6
MFTC 30	-1.0	0.3	3.7	16.2	24.8	117.8	-14.0	6.1	232.7	-22.0	1.6	112.0

Table 1: Percentage cost increase (% Incr) with respect to the `Greedy-alone` algorithm alone, and computing time (in CPU sec.s); a negative increase corresponds to a better solution w.r.t. `Greedy-alone`. The reported values are arithmetic means over 5 random instances. Column % STD gives the percentage standard deviation of cost increase. Computing times for `Greedy-alone` are just negligible. Superscript ^{k} means proven optimality for k out of 5 instances (proof of optimality, when available, being obtained by any of the exact methods).

According to Table 1, `Greedy+Polish` outperforms its competitors by a large amount.

As expected, `Greedy-alone` is always able to provide feasible solutions in very short computing times. In spite of its greedy nature, the solution quality is fair in many cases, in particular in the easiest scenarios where the greedy solution often turns out to be optimal. Nevertheless, for more difficult scenarios there is room for large improvements—also because of the contribution of batteries that is exploited by the MIP model but not by the greedy.

As to `Cplex`, it has a great difficulty even in finding its first feasible solution—a task that takes a huge amount of time in the difficult cases. Significantly improved solutions are found by `Cplex+Polish`, thus confirming the effectiveness of this heuristic. However, the full power of MIP refinement is only exploited when `Greedy+Polish` comes into play. This is due to two main factors: the speed of the greedy, and the fact that several diversified solutions are passed to the polishing method.

Of course, we cannot claim that `Greedy+Polish` would outperform more sophisticated heuristic approaches from the literature on similar problems—for that, much more extensive computational comparisons would be needed. However, we believe our computational results support the message of the present paper—sound matheuristics can be built around a simple greedy and an off-the-shelf MIP refinement procedure.

6 Conclusions

A simple MIP-and-refine matheuristic framework has been addressed, where a greedy heuristic is used to trigger a general purpose MIP refinement procedure. Computational results on a smart-grid energy management problem have been presented, showing that the method produces sound results.

The approach is based on two ingredients: an initial heuristic, and a MIP model. The heuristic needs not to be very effective, as its role is just to initialize a pool of feasible solutions–the more diversified the better. The MIP model itself needs not to be very sophisticated, as it is automatically

resized by the general purpose MIP refinement procedure. Nevertheless, the combination of the two can be much more effective than the sum of its parts, in the sense that the two modules work in a highly synergic way and can produce outcomes whose solution quality can only be matched by sophisticated ad-hoc heuristics.

Future research on the topic will hopefully confirm the viability of the approach on other classes of very difficult problems.

Acknowledgements

The research of the first author was supported by the *Progetto di Ateneo* on "Computational Integer Programming" of the University of Padova, and by MiUR, Italy (PRIN project "Integrated Approaches to Discrete and Nonlinear Optimization").

References

[1] A. Agnetis, G. Dellino, P. Detti, G. Innocenti, G. De Pascale, and A. Vicino. Appliance operation scheduling for electricity consumption optimization. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), Orlando, FL, USA, December 12-15*, pages 5899–5904, 2011.

[2] A. Barbato, A. Capone, M. Carello, Delfanti, M. Merlo, and A. Zaminga. House energy demand optimization in single and multi-user scenarios. In *2nd IEEE International Conference on Smart Grid Communications*, 2011.

[3] A Barbato and G. Carpentieri. Model and algorithms for the real time management of residential electricity demand. In *IEEE International Conference and Exhibition, ENERGYCON '12*, 2012.

[4] G. Carpentieri, G. Carello, and E. Amaldi. Optimization models for residential energy load management. In *Atti delle Giornate AIRO, Vietri*, 2012.

[5] X. Fang, S. Misra, G. Xue, and D. Yang. Smart grid – the new and improved power grid: A survey. *IEEE Communications Surveys and Tutorials (COMST)*, 14:944–980, 2012.

[6] S. Hatami and M. Pedram. Minimizing the electricity bill of cooperative users under a quasi-dynamic pricing model. *SmartGridComm10*, pages 421–426, 2010.

[7] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.

[8] G. Sartor. Optimal scheduling of smart home appliances using mixed-integer linear programming. *Master's Thesis, DEI, University of Padova*, 2012.

[9] K. C. Sou, J. Weimer, H. Sandberg, and K. H. Johansson. Scheduling smart home appliances using mixed integer linear programming. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), Orlando, FL, USA, December 12-15*, pages 5144–5149. IEEE, 2011.

[10] D. Zhang, L.G. Papageorgiou, N.J. Samsatli, and N. Shah. Optimal scheduling of smart homes energy consumption with microgrid. In *ENERGY 2011, The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 70–75, 2011.