Load Disaggregation Based on Aided Linear Integer Programming

Md. Zulfiquar Ali Bhotto, Stephen Makonin, and Ivan V. Bajić

***Abstract*—Load disaggregation based on aided linear integer programming (ALIP) is proposed. We start with a conventional linear integer programming (IP) based disaggregation and enhance it in several ways. The enhancements include additional constraints, correction based on a state diagram, median filtering, and linear programming-based refinement. With the aid of these enhancements, the performance of IP-based disaggregation is significantly improved. The proposed ALIP system relies only on the instantaneous load samples instead of waveform signatures, and hence works well on low-frequency data. Experimental results show that the proposed ALIP system performs better than conventional IP-based load disaggregation.**

***Index Terms*—Integer programming, combinatorial optimization, linear programming, load disaggregation, NILM**

I. Introduction

Load disaggregation or non-intrusive load monitoring (NILM) is the process of finding out how much each appliance within a household is consuming when only the aggregate current or power reading is available [1]. Such readings are now available through smart meters, which have been, or are being, installed by most power utilities. In addition to determining appliance consumption patterns, NILM could help balance different loads within a power network [2] by predicting demand without the use of additional sensors.

Recent disaggregation methods make use of machine learning approaches such as clustering [3], fuzzy systems [4], and hidden Markov models [5]–[9]. Such methods might lead to practical solutions when large and sufficiently representative datasets become available for training, which is still not the case. The ultimate goal for NILM is to enable disaggregation without the need for supervised learning. There has been some recent progress in this area [10], [11], although the accuracy of such methods is still low. Alternative approaches such as combinatorial optimization or integer programming (IP) have been much less explored. Two notable earlier works on IP-based disaggregation include Egarter *et al.* [12] and Suzuki *et al.* [13]. Egarter *et al.* formulated disaggregation as a modified knapsack problem and proposed a solution using an evolutionary algorithm. From a practical standpoint, the drawback here is that evolutionary algorithms potentially have a long execution time, and their stochastic nature may lead to different solutions in different runs, even on the same data. Our goal here is to develop a simpler and more principled approach that gives repeatable results.

The other IP-based disaggregation proposal was by Suzuki *et al.* [13] in 2008 using high-frequency sampled current readings. We extract the IP portion of [13] (without the load signature part) and enhance it in multiple ways to improve disaggregation accuracy. The enhancements include additional constraints, correction based on a state diagram, median filtering, and linear programming-based refinement.Our method works on low-frequency data, which is a more realistic solution for current smart meters that usually report power readings at intervals of 8–15 seconds[1].

The remainder of the paper is organized as follows. We mathematically define the problem of load disaggregation as a mixed-integer linear IP problem in Section II. We propose several enhancements to IP-based disaggregation in Section III, which is our main contribution. A number of experimental results with a comparison to previous works are reported in Section IV, followed by conclusions in Section V.

II. Load Disaggregation as Integer Programming

Consider a household with n appliances, where the i-th appliance ($i = 1, 2, \ldots, n$) has l_i non-OFF states. For example, a conventional light bulb would have one non-OFF state. Vector $\mathbf{r}_i \in \mathbb{R}^{l_i}$ contains the voltampere (VA) ratings of the l_i non-OFF states of the i-th appliance. Let $m = \sum_{i=1}^{n} l_i$. With this we construct vector $\mathbf{r} = [\mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_n]$ that contains all VA (or Watt) ratings of all n appliances. For the k-th time instant, the indicator of each non-OFF state is stored in a vector $\mathbf{b}_k$ as

$$\mathbf{b}_k[i] \in \{1,\ 0\} \text{ for } i = 1, 2, \ldots, m, \tag{1}$$

with 1 denoting that the particular state is active and 0 denoting the state is inactive.

At the k-th time instant, the smart meter yields the total VA reading z_k, which is the sum of all power drawn by n appliances at that time. This can be expressed as

$$\mathbf{s}_k = \mathbf{F}\ \mathtt{diag}(\mathbf{b}_k)\mathbf{r} \tag{2}$$

$$z_k = \mathbf{h}^t \mathbf{s}_k \tag{3}$$

where $\mathbf{h} = [1;\ 1;\ \cdots;\ 1] \in \mathbb{R}^n$ and vector $\mathbf{s}_k \in \mathbb{R}^n$ contains VA draws of each appliance that is turned ON. The binary matrix $\mathbf{F} \in \mathbb{R}^{n \times m}$ in (2) is a block diagonal matrix given as

$$\mathbf{F} = \mathtt{diag}(\mathbf{1}_1^t,\ \mathbf{1}_2^t,\ \ldots, \mathbf{1}_n^t)$$

Manuscript received Mar. 2016, revised Jun. and Jul. 2016. This work was supported by the Natural Sciences and Engineering Research Council of Canada under Grant EGP-486210-15.

The authors are with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: mbhotto@sfu.ca, smakonin@sfu.ca, ibajic@ensc.sfu.ca).

[1]For example, Rainforest Automation's EMU2 device polls the smart meter at 15 second intervals, while their Eagle product polls at 8 seconds. In the UK it is mandated to 10 seconds [14, p. 78].

where $\mathbf{1}_i \in \mathbb{R}^{l_i}$ is a unity vector for $i = 1, 2, \ldots, n$.

The objective of load disaggregation is to find which appliance states are active at the k-th time instant. Specifically, the goal is to find $\mathbf{b}_k$ in (2) by using the known quantities z_k, $\mathbf{F}$, and $\mathbf{r}$. Similarly to [13], load disaggregation can be formulated as an integer programming (IP) problem

$$\underset{\mathbf{b}_k}{\text{minimize}} \quad \left(z_k - \mathbf{h}^t \mathbf{F}\ \texttt{diag}(\mathbf{b}_k)\ \mathbf{r}\right)^2. \tag{4}$$

Like any IP, this problem can be solved by exhaustive search over all possibilities for $\mathbf{b}_k$, however this approach can be prohibitive even for a moderate number of appliances. The alternative is to explore more efficient IP solvers [15].

Before proceeding to enhancements, we reformulate (4) as a mixed-integer linear IP. Let $q_j = \sum_{i=1}^{j-1} l_i$. Since any appliance must be in exactly one state at any given time and vector $\mathbf{b}_k$ is an indicator for non-OFF states, we can formulate this constraint as

$$\mathbf{b}_k[q_j + 1] + \mathbf{b}_k[q_j + 2] + \cdots + \mathbf{b}_k[q_j + l_j] \leq 1$$

for $j = 1, 2, ..., n$. Let $\mathbf{v} = \mathbf{r} \odot (\mathbf{F}^t \mathbf{h})$ where $\odot$ denotes the element-wise product. Further, let $\check{\mathbf{v}} = [0;\ \mathbf{v}]$, $\mathbf{u}_1 = [1;\ \mathbf{0}]$, $\mathbf{f} = [1;\ \mathbf{0}]$, and $\mathbf{x}_k = [\delta;\ \mathbf{b}_k]$, where δ is an auxiliary real variable. We can now rewrite the quadratic IP in (4) as a mixed-integer *linear* IP

$$\underset{\mathbf{x}_k}{\text{minimize}} \quad \mathbf{f}^t \mathbf{x}_k \tag{5}$$

subject to

$$\mathbf{A}\mathbf{x}_k \quad \leq \quad \mathbf{e} \tag{6}$$

$$\mathbf{x}_k[i] \quad \in \quad \{1, 0\} \text{ for } i = 2, 3, \ldots, m+1 \tag{7}$$

where matrix $\mathbf{A}$ and vector $\mathbf{e}$ in (6) are given by $\mathbf{A} = [-(\check{\mathbf{v}} + \mathbf{u}_1)^t;\ (\check{\mathbf{v}} - \mathbf{u}_1)^t;\ \check{\mathbf{F}}]$ and $\mathbf{e} = [-z_k;\ z_k;\ \mathbf{1}]$ and the rows of the binary matrix $\check{\mathbf{F}}$ are copies of the rows of matrix $\mathbf{F}$ that have more than one nonzero element.

The solution of (5)–(7) leads to correct disaggregation only if the elements in $\mathbf{r}$ are not binary combinations (linear combinations with coefficients 0 or 1) of each other, and the deviations from the steady-state power/current draw during transients do not overlap with the steady-state draws of other appliances, or their combinations. In practice, these criteria are rarely met, so disaggregation based on (5)–(7) would yield unsatisfactory results. This can also be true for an appliance with an infrequently occurring state with a high rating, since an undetected state with high rating would significantly reduce the accuracy score (Section IV). In the next section we introduce several enhancements that are meant to overcome the aforementioned limitations of the above IP disaggregation.

III. Aided Linear IP for Load Disaggregation

The proposed aided linear IP (ALIP) for load disaggregation incorporates several enhancements to the IP given in (5)–(7), each of which is discussed next.

A. Additional Constraints

The first enhancement involves additional constraints that help resolve ambiguities related to the possible non-uniqueness of the solution to the IP. First, consider appliances like refrigerator, surveillance camera, smoke detector, heat pump, etc., that happen to switch between the "sleep mode" and one or more higher-power states. In other words, these appliances always draw some power. We can incorporate this additional information as an equality constraint to be added to (5)–(7),

$$\mathbf{A}_{eq}\mathbf{x}_k = \mathbf{1} \tag{8}$$

where each row in the binary matrix $\mathbf{A}_{eq}$ has unity elements only in those positions that correspond to the states of those appliances in vector $\mathbf{x}_k$ that remain turned ON at all time.

Next, consider the scenario where the rating of a given state of an appliance is equal to the sum of the ratings of some of the states of other appliances. For example, an appliance H1 could have a VA rating 300 in one of its states, and appliances H2 and H3 could have VA ratings of 100 and 200, respectively, in some of their states. Then a reading $z_k = 300$ could be interpreted in two ways - either H1 is ON, or H2 and H3 are simultaneously ON. To break such ties, we assume the minimum number of appliances is ON at any given time - a heuristic that does not always hold, but turns out to be surprisingly good based on empirical evidence in existing datasets. This can be incorporated into the IP by using an additional row in the binary $\mathbf{A}$ and an additional 1 in $\mathbf{e}$ in (6). The additional row in $\mathbf{A}$ would have unity elements only in those positions that correspond to the states where the ratings become binary combinations of each other.

Finally, consider the scenario where the rating of a given appliance (say H1) is equal to the amount by which the steady-state rating of another appliance (say H2) differs from its transient VA measurement. The transient reading may cause the IP solver to declare both H1 and H2 as ON, even though only H2 is in the ON transient. Such situation can also be avoided by using an additional row in the binary matrix $\mathbf{A}$ and an additional 1 in $\mathbf{e}$ in (6), where the additional row in $\mathbf{A}$ has unity elements only in those positions that correspond to the states whose combinations produce a transient rating of another appliance.

B. Correction Based on State Transition Diagrams

Many appliances operate as finite state machines and their possible state transitions can be described by a state transition diagram (STD). For example, the fridge (FRG) appliance from the dataset in [16] has the STD shown in Fig. 1. This offers the possibility to correct the output of an IP solver if it happens to violate the STD. For example, if FRG was in state s_1 at time $k-1$, then at time k it can only be in s_1 or s_2. If the IP solver output suggests otherwise, we know there must be an error, and can therefore select either s_1 or s_2, depending on which of the two options yields lower cost $\mathbf{f}^t\mathbf{x}_k$ in (5). The same type of correction can also be applied backwards (for example, the only way to get to s_3 is either from s_2 or s_3), although we did not incorporate such processing in our experiments.

C. Median Filtering

Median filtering can help filter out implausible events such as frequent switching between states, which may occur in the

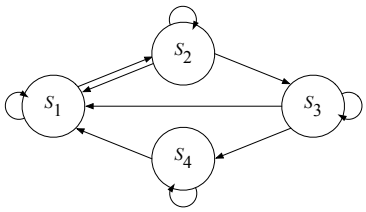

Fig. 1. State transition diagram (STD) of FRG from [16].

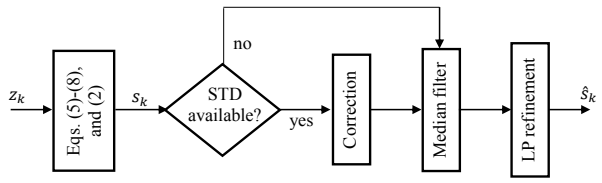

Fig. 2. ALIP flowchart.

IP solver output if the ratings of a particular appliance are much smaller than the total reading z_k. Consider the appliance B1E from [16], which has a fully-connected 2-state diagram (Fig. 1 with states s_3 and s_4 deleted). Although any transition between these two states is possible, it is implausible that the appliance changes state at each sampling instant; we expect it to stay in a state for at least a few sampling instants. To enforce this, we apply the correction rule

$$\hat{s}_{k-L} = \begin{cases} s_1 \text{ if } \hat{s}_{k-L} = s_2 \text{ and } \text{med}(\hat{s}_k, \hat{s}_{k-1}, \ldots, \hat{s}_{k-L}) = s_1 \\ s_2 \text{ if } \hat{s}_{k-L} = s_1 \text{ and } \text{med}(\hat{s}_k, \hat{s}_{k-1}, \ldots, \hat{s}_{k-L}) = s_2 \end{cases} \tag{9}$$

for $k > L$ in the solution obtained by the IP solver. In other words, the state estimate at time $k - L$ (i.e., $\hat{s}_{k-L}$) is corrected based on the current state estimate $\hat{s}_k$ and the corrected state does not affect the subsequent median filter outputs. Analogous corrections are applied to all states of all appliances.

D. Linear Programming-Based Refinement

As mentioned before, vector $\mathbf{r}$ contains steady-state appliance ratings, which could be obtained from appliance data sheets or measurements. However, VA (or Watt) values for transients between states are usually much more difficult to obtain, and even if this were possible, incorporating transient state ratings into the model would tremendously increase the model size (i.e., number of states) m. Yet, if the sampling instance happens to catch a transient of one or more appliances, it could lead to incorrect solution of the IP. For this reason, we develop a method to refine the IP solution using only the minimum and maximum transient rating of each appliance, which is easier to obtain, either from the data sheet or measurement.

Let $\mathbf{r}_{\min}$ and $\mathbf{r}_{\max}$ be the vectors of the same size as $\mathbf{r}$ that contain, respectively, the minimum and maximum transient ratings for each state of each appliance. Let vector $\mathbf{p}_1$ contain indices of $\mathbf{r}$ for which $\mathbf{r}_{\min}(\mathbf{p}_1) = \mathbf{r}_{\max}(\mathbf{p}_1) = \mathbf{r}(\mathbf{p}_1)$. Such states do not exhibit transient behavior. Let the vector $\mathbf{p}_2$ contain the indices of other, potentially transient, states, that have been declared active by the IP solver (i.e., the corresponding value in $\mathbf{b}_k$ is 1). If $\mathbf{p}_2$ is non-empty, then the current measurement may contain transient states and the solution given by the IP solver needs to be refined.

Let $\boldsymbol{h} = \mathbf{F}^t\mathbf{h}$ and $\mathbf{w}_k = \mathbf{b}_k \odot \mathbf{r}$. To refine the solution, we solve the following problem

$$\underset{\mathbf{y}_k}{\text{minimize}} \quad \left(z_k - \boldsymbol{h}^t\mathbf{y}_k\right)^2 \tag{10}$$

subject to

$$\mathbf{y}_k(\mathbf{p}_1) = \mathbf{w}_k(\mathbf{p}_1) \tag{11}$$

$$\mathbf{r}_{\min}(\mathbf{p}_2) \le \mathbf{y}_k(\mathbf{p}_2) \le \mathbf{r}_{\max}(\mathbf{p}_2) \tag{12}$$

Constraints (11)–(12) force the non-transient states to match the steady-state ratings in $\mathbf{r}$ while requiring potentially transient states to be between the corresponding minimum and maximum.

The cost function can be simplified by subtracting out the steady-state portion of the measurement, $\check{z}_k = z_k - \boldsymbol{h}(\mathbf{p}_1)^t\mathbf{w}_k(\mathbf{p}_1)$, and focusing on the transient portion of $\mathbf{y}_k$, i.e., $\check{\mathbf{y}}_k = \mathbf{y}_k(\mathbf{p}_2)$. Then, applying a similar procedure as in Section II, the problem can be converted to a linear programming problem. With $\check{\mathbf{x}} = [\delta;\ \check{\mathbf{y}}_k]$, $\check{\boldsymbol{h}} = [0;\ \mathbf{1}]$, $\mathbf{u}_1 = [1;\ \mathbf{0}]$, and $\mathbf{f} = [1;\ \mathbf{0}]$, the problem becomes

$$\underset{\check{\mathbf{x}}}{\text{minimize}} \quad \mathbf{f}^t\check{\mathbf{x}} \tag{13}$$

subject to

$$\mathbf{A}\check{\mathbf{x}} \le \mathbf{e} \tag{14}$$

where matrix $\mathbf{A}$ and vector $\mathbf{e}$ in (14) are given by $\mathbf{A} = [-(\check{\boldsymbol{h}} + \mathbf{u}_1)^t;\ (\check{\boldsymbol{h}} - \mathbf{u}_1)^t;\ [\mathbf{0}\ -\mathbf{I}];\ [\mathbf{0}\ \mathbf{I};]]$ and $\mathbf{e} = \left[-\check{z}_k;\quad \check{z}_k;\quad -\check{\mathbf{r}}_l;\quad \check{\mathbf{r}}_u\right]$, respectively.

The flowchart of ALIP is shown in Fig. 2. Further steps such as time-of-day probabilities can be incorporated in order to further improve disaggregation accuracy.

IV. Experimental Results

We compare the performance of our ALIP method with the IP-based disaggregation from [13] in terms of two accuracy measures [17]: per appliance accuracy

$$\text{AC}_i = 1 - \frac{\sum_{k=1}^{N} |\mathbf{s}_k[i] - \hat{\mathbf{s}}_k[i]|}{2\sum_{k=1}^{N} |\mathbf{s}_k[i]|}$$

and overall accuracy

$$\text{ACC} = 1 - \frac{\sum_{k=1}^{N}\sum_{i=1}^{n} |\mathbf{s}_k[i] - \hat{\mathbf{s}}_k[i]|}{2\sum_{k=1}^{N}\sum_{i=1}^{n} |\mathbf{s}_k[i]|}$$

where $\mathbf{s}_k[i]$ is the ground-truth rating of the i-th appliance at time index k from the dataset and $\hat{\mathbf{s}}_k[i]$ is its estimate obtained by disaggregation. In all experiments, the steady-state ratings of each appliance were determined empirically from the datasets from the power consumption of each appliance separately. The maximum and minimum transient VA or Watt values for the ALIP disaggregator were also determined empirically. The aggregate VA or Watt totals for the appliances used in the experiments were computed from aggregated individual appliance readings. For ALIP, the enhancements were applied in the order depicted in the flowchart in Fig. 2. All ratings

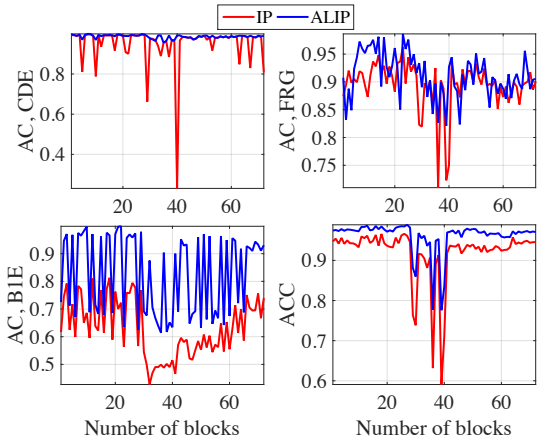

Fig. 3. AC and ACC plots for Experiment 1.

and parameters used in the experiments can be found in the Matlab code[2], which can be used to reproduce all the results.

In Experiment 1 we used $n = 4$ appliances (CDE, FRG, HPE, and B1E) from the dataset in [16]. The number of states considered for the CDE, FRG, HPE, and B1E appliances were 3, 4, 4, and 2, respectively. The total number of samples considered was 72×5040, which we partitioned into 72 blocks of data each containing $N = 5040$ samples. This number of samples covers 9 months worth of readings. We computed AC and ACC for each block. The AC curves obtained by using the IP and proposed ALIP disaggregators for CDE, FRG, and B1E appliances are illustrated in Fig. 3, along with the ACC curves. We have not illustrated AC curves for HPE, since both disaggregators produced very similar curves for this appliance.

It is seen from the AC, CDE plots (top-left in Fig. 3) that ALIP performs better than IP consistently in all blocks. The same is true for B1E (bottom-left in Fig. 3). From the AC, FRG plots it is seen that the ALIP performs considerably better than IP in many blocks, while IP performs marginally better in some blocks. This is because FRG has occasional impulsive VA readings which get filtered out by ALIP but not by IP. Overall, however, ALIP disaggregates FRG better than IP. From the ACC plots (bottom-right of Fig. 3), it can be seen the ALIP disaggregator is overall more accurate than the IP disaggregator - usually by 5–8%, and in some cases by up to 20%. The AC values obtained over the whole 9 months worth of data are given in Table I, while the overall ACC measure for this and other experiments is given in Table II. As seen in these tables, ALIP outperforms IP on each appliance, as well as overall.

In Experiment 2 we used $n = 7$ appliances (OVN, RFG, DSH, MIC, DRY, BTH, and DIF) from house 1 of the REDD dataset [5], with $m = 13$ states. We down-sampled the data by a factor 20 to obtain the samples at 1-minute intervals. The AC values are given in Table I and the overall ACC in Table II. Again, ALIP performs better than IP on all appliances individually, as well as overall.

In Experiment 3 we used $n = 6$ appliances (KTC, LTE, STV, MIC, DRY, and DSH) from house 2 in [5], with $m = 17$ states. We downsampled the data by a factor 5 to obtain the samples at 15-second intervals. The AC values are given in Table I. As seen in the results, ALIP performs significantly better than IP on all appliances except DRY. This is because DRY has a state with a high VA rating, whose occurrence is infrequent, and the median filter has filtered out some of its occurrences. As a result, the accuracy of ALIP gets reduced compared to IP on DRY. Nonetheless, the accuracy of ALIP on DRY is still acceptable and its overall accuracy is significantly better than that of IP, as seen in Table II.

In Experiment 4 we used $n = 7$ appliances (KTC, LTE, STV, MIC, DRY, and DSH) from house 3 in [5], with $m = 20$ states. We downsampled the data by a factor 10 to obtain the samples at 30-second intervals. The AC values are given in Table I, where we see that ALIP performs significantly better than IP on all appliances except FRG, where it performs similarly due to the reasons discussed in Experiment 1. Overall, again, ALIP outperforms IP by a significant margin as Seen in Table II.

In Experiment 5 we used $n = 8$ appliances (LTE, KTC, DRY, STV, ARC, SMK, DSH, and BTH) from house 4 in [5],

[2]http://www.sfu.ca/~ibajic/software/NILM-TCAS.rar

TABLE I
RESULTS FOR EXP. 1–7

Exp.	Dataset	ID	Appliance	IP	ALIP
1	AMPds	CDE	Clothes Dryer	0.986	0.987
		FRE	Furnace Fan	0.891	0.909
		HPE	Heat Pump	0.955	0.957
		B1E	Bedroom	0.600	0.757
2	REDD	OVN	Oven	0.65	0.63
	House 1	FRG	Refrigerator	0.79	0.85
		DSH	Dishwasher	0.88	0.92
		MIC	Microwave	0.74	0.83
		DRY	Clothes Dryer	0.64	0.78
		BTH	Bathroom GFI	0.64	0.70
		DIF	Unmetered	0.60	0.95
3	REDD	KTC	Kitchen Outlets	0.88	0.95
	House 2	LTE	Lighting	0.91	0.96
		STV	Stove	0.71	0.75
		MIC	Microwave	0.78	0.88
		DRY	Clothes Dryer	0.98	0.85
		DSH	Dishwasher	0.84	0.90
4	REDD	FRG	Refrigerator	0.91	0.89
	House 3	DSH	Dishwasher	0.83	0.90
		DRY	Clothes Dryer	0.83	0.99
		MIC	Microwave	0.59	0.87
		BTH	Bathroom GFI	0.68	0.91
		FRN	Furnace	0.66	0.76
		SMK	Smoke Detector	0.16	0.64
5	REDD	LTE	Lights	0.75	0.73
	House 4	KTC	Kitchen Outlets	0.64	0.80
		DRY	Clothes Dryer	0.67	0.65
		STV	Stove	0.64	0.66
		ARC	Air Conditioner	0.57	0.53
		SMK	Smoke Detector	0.32	0.79
		DSH	Dishwasher	0.70	0.88
		BTH	Bathroom GFI	0.82	0.89
6	REDD	MIC	Microwave	0.38	0.55
	House 5	LTE	Lighting	0.77	0.84
		UKN	Unknown Circuit	0.61	0.67
		SBP	Sub-Panel	0.62	0.72
		HEA	Heater	0.91	0.93
		DIF	Unmetered	0.75	0.96
7	REDD	ELE	Electronics	0.70	0.70
	House 6	BTH	Bathroom GFI	0.53	0.56
		FRG	Refrigerator	0.64	0.77
		UKN	Unknown Circuit	0.68	0.73
		LTE	Lighting	0.64	0.91
		ARC	Air Conditioner	0.73	0.98
		DIF	Unmetered	0.51	0.70

with $m = 20$ states. The data was downsampled factor 10 to obtain the samples at 30-second intervals. As seen in Table I, IP performs slightly better than ALIP on LTE, DRY and ARC, but the difference is small, within 4%. However, on other appliances ALIP performs better than IP, and in most cases by a significant margin. Note that IP yields a negative AC value for SMK, which means that it produces false positives more often than true positives. Overall, ALIP has a significant advantage over IP, as seen in Table II.

TABLE II
ACC RESULTS (INCL. PUBLISHED COMPARISON)

	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7
IP	0.92	0.51	0.70	0.64	0.43	0.57	0.46
ALIP	**0.96**	0.76	0.82	**0.87**	**0.76**	**0.83**	**0.79**
[5]	–	0.47	0.51	0.33	0.52	–	0.56
[6]	–	**0.82**	**0.85**	0.82	–	–	0.78

TABLE III
AVERAGE EXECUTION TIME PER DATA SAMPLE (MILLISECONDS)

	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Exp. 6	Exp. 7
IP	14.2	15.3	15.1	14.7	16.0	17.0	17.3
ALIP	15.4	16.7	17.0	16.9	18.5	17.0	18.8

In Experiment 6 we used $n = 6$ appliances (MIC, LTE, UKN, SBP, HEA, and DIF) from house 5 in [5], with $m = 24$ states. The data were downsampled by a factor 10 to obtain the samples at 30-second intervals. Here, ALIP outperforms IP on all appliances, as well as in overall accuracy, as seen in Tables I and II, respectively.

Finally, in Experiment 7 we used $n = 7$ appliances (ELE, BTH, FRG, UKN, LTE, ARC, and DIF) from house 6 in [5], with $m = 20$ states. The data were downsampled by a factor 10 to obtain the samples at 30-second intervals. Again, ALIP outperforms IP consistently on all appliances, as well as in overall accuracy, as seen in Tables I and II, respectively.

In Table II we also include the published ACC results of two state-of-the-art machine learning-based disaggregation approaches, [5] and [6], on the REDD dataset. Note that [5] and [6] did not report the results for all the houses. Although these methods used different downsampling rates and there is some uncertainty about the processing of data prior to testing, the comparison gives us a feeling for how the proposed ALIP would compare against machine learning-based disaggregation. We note that against [5], ALIP scores on average 0.32 better. Against [6], it scores on average 0.03 better on houses 3 and 6, and on average 0.04 lower on houses 1 and 2. Based on these results, we conclude that the proposed ALIP is competitive in terms of accuracy with the state-of-the-art machine learning-based disaggregation approaches.

A final word on complexity. The proposed ALIP approach incorporates several additional processing steps compared to the conventional IP-based disaggregation. Hence, its computational complexity is slightly higher. In Table III we list the average execution time (in milliseconds) per sample of IP and ALIP disaggregators for Experiments 1–7, which were obtained in Matlab 2015a using `intlinprog` and `linprog` (with default settings) on an Intel(R) Core(TM) i7-4770 CPU@3.40 GHz processor with 16 GB RAM. As seen in the table, ALIP is only slightly slower than IP, and both disaggregators take less than 20 ms per data sample.

V. CONCLUSION

Integer programming (IP) provides a natural way to solve the load disaggregation problem, by trying to determine which appliance states are active at any given time. However, previous IP-based disaggregation is shown to run into problems on real data due to issues related to transient readings and in cases when some states are binary combinations of other states. We proposed an aided linear IP (ALIP) approach for disaggregation that overcomes many of the shortcomings of the previous IP-based approach. Experimental results demonstrate significant accuracy advantage of ALIP over the previous IP-based disaggregation method, as well as competitive performance against two state-of-the art machine learning-based disaggregation approaches.

REFERENCES

[1] S. Makonin, F. Popowich, and B. Gill, "The cognitive power meter: Looking beyond the smart meter," in *Proc. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2013.

[2] V. Zdraveski, M. Todorovski, D. Trajanov, and L. Kocarev, "Dynamic load balancing and reactive power compensation switch embedded in power meters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2016.

[3] G. C. Koutitas and L. Tassiulas, "Low cost disaggregation of smart meter sensor data," *IEEE Sensors J.*, vol. 16, pp. 1665–1673, Mar. 2016.

[4] P. Ducange, F. Marcelloni, and M. Antonelli, "A novel approach based on finite-state machines with fuzzy transitions for nonintrusive home appliance monitoring," *IEEE Trans. Ind. Informat.*, vol. 10, pp. 1185–1197, May 2014.

[5] J. Z. Kolter and M. J. Johnson, "REDD: a public data set for energy disaggregation research," in *Proc. SustKDD Workshop on Data Mining Applications in Sustainability*, 2011.

[6] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric hidden semi-markov models," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 673–701, 2013.

[7] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *IEEE Trans. Consum. Electron.*, vol. 58, pp. 23–31, Feb. 2012.

[8] D. Egarter, V. P. Bhuvana, and W. Elmenreich, "PALDi: online load disaggregation via particle filtering," *IEEE Trans. Instrum. Meas.*, vol. 64, pp. 467–477, Feb. 2015.

[9] S. Makonin, F. Popowich, I. V. Bajić, B. Gill, and L. Bartram, "Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring," *IEEE Trans. Smart Grid*, vol. PP, no. 99, pp. 1–11, 2015.

[10] B. Zhao, L. Stankovic, and V. Stankovic, "On a training-less solution for non-intrusive appliance load monitoring using graph signal processing," *IEEE Access*, vol. 4, pp. 1784–1799, 2016.

[11] D. Egarter and W. Elmenreich, "Autonomous load disaggregation approach based on active power measurements," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pp. 293–298, IEEE, 2015.

[12] D. Egarter, A. Sobe, and W. Elmenreich, "Evolving non-intrusive load monitoring," in *Proc. 16th European Conference on Applications of Evolutionary Computation: EvoApplications'13*, (Vienna, Austria), pp. 182–191, Apr. 2013.

[13] K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura, and K. Ito, "Nonintrusive appliance load monitoring based on integer programming," in *Proc. SICE Annual Conference*, pp. 2742–2747, 2008.

[14] "Smart metering implementation programme," tech. rep., Dept. Energy Climate Change, London, U.K., 2013.

[15] A. Antoniou and W. S. Lu, *Practical Optimization*. Springer, 2007.

[16] S. Makonin, B. Ellert, I. V. Bajić, and F. Popowich, "Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014," *Scientific Data*, vol. 3, no. 160037, pp. 1–12, 2016.

[17] S. Makonin and F. Popowich, "Nonintrusive load monitoring (NILM) performance evaluation," *Energ. Effic.*, vol. 8, no. 4, pp. 809–814, 2015.