An Efficient Screening Method for Identifying Parameters and Interactions that Impact Wireless Network Performance

Violet R. Syrotiuk, Charles J. Colbourn, Eli De Poorter, Michael Tetemke Mehari, and Ingrid Moerman

Dedicated to Doug Stinson, in appreciation of his deep mathematics with genuine applications, on the occasion of his 66th birthday.

Abstract Wireless networks rely on a protocol stack for their operation. Not only are the protocols at each layer configurable, potential interactions arise among the protocol stack, operating system, hardware, and operating environment. Hence, there is a vital need for screening, i.e., to determine the parameters and interactions that significantly impact performance of a system. In this paper, we propose: (1) the use of a locating array (LA) as the design of a screening experiment, and (2) an algorithm to analyze the resulting measured performance data. Compared to conventional designs, LAs grow logarithmically in the number of parameters making them efficient for screening complex engineered networks. The analysis uses a framework from compressive sensing and provides robustness to noise in the system through a breadth-first search strategy. We apply LAs for screening audio quality and radio frequency exposure in a Wi-Fi conferencing scenario in the `w-iLab.t` wireless network testbed, and validate the results using the Dantzig selector and Lasso regression methods.

1 Introduction

Experimentation is a cornerstone of scientific advancement. Through experimentation we gather evidence to either support or refute a hypothesis. A crucial question

Violet R. Syrotiuk and Charles J. Colbourn
School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, USA 85287-8809, e-mail: `\{syrotiuk,colbourn\}@asu.edu`

Eli De Poorter, Michael Tetemke Mehari, and Ingrid Moerman
Department of Information Technology, Internet and Data Science Lab, Ghent University – IMEC, B-9052 Ghent, Belgium, e-mail: `\{eli.depoorter,michael.mehari,ingrid.moerman\}@ugent.be`

is: *Which parameters should be selected for experimentation?* Domain experts often use their knowledge and experience to select these parameters. But when the system is large and complex, a systematic method that could answer this question would increase confidence in the experimental strategy.

In networking, interactions between protocols are known and these interactions are not always between protocols in adjacent layers. For example, a two-way interaction was discovered that involves the *transport control protocol* (TCP) interpreting access delays in a wireless link due to poor signal quality as congestion, and hence responds incorrectly with congestion control [4]. Many such cross-layer interactions are known; see [1, 5, 9, 16, 34, 40] as examples. These are not always evident, even to domain experts. If not certain that a parameter or interaction affects performance, a knowledgeable domain expert may ignore it in order to reduce experimental cost. Yet this guarantees that its impact on the performance is never observed.

In this paper, our interest is to use experimentation to identify the significant parameters and two-way interactions impacting wireless network performance. These are termed *screening experiments* [27]. To cope with the complexity in engineered network systems, screening *should be* an important first step before conducting the intended experiment, such as to optimize system performance, or to improve the robustness of the system to operating conditions.

In the field of *designed experiments*, it is considered impractical to experiment with more than about 10 parameters [18, 27]. Most protocols at each layer of a typical TCP/IP stack have at least 5-10 configurable parameters. Thus, there can be 25-50 parameters to vary in experimentation without considering wider aspects of the system. Indeed, parameters of the operating system (e.g., kernel version, buffer sizes, queuing disciplines), the hardware (e.g., chipset, drivers), and the operating environment may also impact the network performance.

As we will see, for engineered network systems many traditional screening experiments are infeasible. This is because the *experimental design*, an array describing the runs in the experiment, is too large. While supersaturated designs (SSDs) can screen efficiently, their focus is primarily on identifying main effects [12, 19, 27]. In networking, ideally we are interested in a screening method that is also capable of identifying two-way interactions, because some parameters may be significant only as a result of their involvement in a significant interaction.

To address this need, we use a combinatorial design called a *locating array* (LA) for screening. Locating arrays grow logarithmically in the number of parameters [7]. Therefore, they have the potential to screen efficiently a far larger number of parameters and two-way interactions than previous approaches. Not surprisingly, there is a trade-off: One reason locating arrays are small is because they are often unbalanced. Balance relates to how equally a parameter or interaction is measured in the design. Most analysis tools, such as JMP [30], assume the underlying experimental design is balanced, or nearly balanced. Because locating arrays may be highly unbalanced, we are unable to apply the standard analysis techniques to recover the significant parameters and interactions from the performance measurements.

Thus in addition to introducing locating arrays as a screening design, another contribution of this paper is to propose a new analysis technique to accomplish the identification. We use the framework of compressive sensing to recover a model whose terms correspond to the parameters and two-way interactions that significantly impact the performance. However, because measurements of real systems are noisy, *any* recovery approach could make an error in term identification, which could impact the identification of subsequent terms. Rather than recover a single model, we use a search tree to recover a number of alternative models, providing an analyst with flexibility in understanding the performance of a complex system.

Recall that our original motivation is to screen, so that we can conduct a follow-on experiment having confidence that we will be varying the parameters and interactions that affect performance significantly. Because of the efficiency of locating arrays we do not need to reduce the number of parameters considered in screening a priori. Our methodology is a systematic and efficient way to screen large design spaces, thereby reducing the likelihood that significant parameters or two-way interactions are missed in follow-up experimentation.

We apply our proposed design and analysis methodology to screen 22 parameters and their two-way interactions, in a Wi-Fi conferencing scenario set up in the `w-iLab.t` wireless network testbed [3]. The parameters are taken from the kernel's IP and UDP protocols, the Wi-Fi card driver, the audio codec, and a source of radio frequency (RF) interference. The LA used as experimental design has only 73 runs; this compares with $\approx 5.3 \times 10^{12}$, an infeasible number of runs in a full-factorial design. We apply our analysis technique to the measurements collected from `w-iLab.t` to identify the significant parameters and two-way interactions impacting the audio quality and the RF exposure. The results are validated using the Dantzig selector [28] and Lasso regression [37] methods.

To enable reproducibility of our results, we provide all the code, scripts, and tools necessary to construct experimental designs based on LAs, and analyze the measurements collected from experiments based on them [13, 36].

The rest of this paper is organized as follows. §2 overviews traditional designs used for screening and their analysis, and provides the definition of a LA. §3 presents the proposed analysis methodology. This is followed by the details of the experimental set-up in §4, and the results of applying the analysis methodology to the performance measurements, and their validation, in §5. Finally, we conclude in §6, suggesting several opportunities for future research.

2 Screening Designs

2.1 Definition of a Run, an Experiment, and a Design

Suppose that the system under study has k parameters, $P_1, \ldots, P_k$, and that each parameter P_j has a set $V_j = \{v_{j,1}, \ldots, v_{j,\ell_j}\}$, of ℓ_j possible values. A *run* is an assignment of a value from V_j to P_j, for each parameter $j = 1, \ldots, k$. An *experimental design* (or, design for short) is a collection of runs.

When a design has *size* N, it is represented by an $N \times k$ array $A = (a_{ij})$ in which each row i corresponds to a run and each column j to a parameter; the entry a_{ij} specifies the value assigned to parameter j in the ith run. When executed on a system, a run results in the measurement of one or more performance metrics. An *experiment* consists of executing each run in the design.

2.2 A Running Example

We introduce an example to explain properties of experimental designs. Consider a system with 4 parameters. Parameters A and B each have two values $V_A = V_B = \{0, 1\}$, while parameters C and D each have three values $V_C = V_D = \{0, 1, 2\}$. For short, we use the notation P_ℓ to denote parameter P set to the value equal to ℓ. For example, A_1 denotes A set to 1.

2.3 Traditional Screening Designs

A *full-factorial design* has runs that include all possible combinations of values of each parameter P_j across all k parameters [8]. Its size is equal to the product of $|V_j|$ for each parameter $j = 1, \ldots, k$. For the running example, a full-factorial design has $2^2 \times 3^2 = 36$ runs, and in general the number of runs grows exponentially in the number of parameters. From an *analysis of variance* (ANOVA) of the data collected from a full-factorial design all significant t-way interactions for $t = 1, \ldots, k$ can be identified. Traditionally, identifying significant main effects and two-way interactions, i.e., $t = 1, 2$, have been of primary interest, as higher-order interactions tend to be rare and of lesser effect [21, 27].

More recently, *supersaturated designs* (SSDs) have been introduced to identify significant main effects [12]. This focus comes from relying on an assumption of *strong heredity*, the condition that a significant two-way interaction has its component main effects also significant. However, strong heredity is not universally valid in real-world experiments [21].

Many criteria are used to optimize supersaturated designs, e.g., to obtain more confidence in the parameters that are identified [17]. One popular one, *D-optimality*, minimizes the size of the joint confidence region for the model coefficients. Supersaturated designs employ advanced analysis methods [19].

A problem with most traditional screening designs is that they do not ensure it is possible to estimate the effects of all interactions, or even that they all occur in the design. If a significant assignment of values to parameters is missing from a design, it is impossible to determine this from the data collected in the experiment. Covering arrays aim to address this issue.

2.4 Covering Arrays

Covering problems have been studied extensively in combinatorial design theory [35]. An assignment of values to any subset $t \leq k$ of the parameters is a *t-way interaction*. A *covering array* (CA) of strength t, is an $N \times k$ array in which for every $N \times t$ subarray, each t-way interaction is *covered* (i.e., occurs) in at least one run [14]. A covering array of strength two on the four parameters of the running example is given in Table 1(a). Nine runs suffice to cover all 37 of the two-way interactions; e.g., $A_0 C_2$ is covered in run 5.

Table 1 For the running example: (a) A CA C of strength 2; (b) a (1, 2)-LA L

(a)

Run	A	B	C	D
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	1	2
5	0	1	2	1
6	1	0	2	2
7	1	1	0	2
8	1	1	1	1
9	1	1	2	0

(b)

Run	A	B	C	D
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	1	1
5	0	0	2	2
6	0	1	0	2
7	0	1	1	2
8	0	1	2	0
9	0	1	2	1
10	1	0	0	2
11	1	0	1	2
12	1	0	2	0
13	1	0	2	1
14	1	1	0	0
15	1	1	0	1
16	1	1	1	0
17	1	1	1	1
18	1	1	2	2

Covering arrays have recently been introduced as experimental designs into the software tool JMP [30]. Analysis is simplified when a design is *balanced*. In general, covering arrays can be close enough to balanced for the traditional approaches for analysis to succeed.

While a covering array of strength t ensures coverage, it does not ensure that it is possible to distinguish the influence of different t-way interactions. For example, if the performance metric measured for run 5 is different from the other runs, it is not possible to determine which of the three two-way interactions $A_0 B_1$, $A_0 C_2$, and $C_2 D_1$ is responsible, because each one appears only in run 5. Locating arrays extend covering arrays to address this very issue.

2.5 Locating Arrays

A (d, t)-*locating array* [7] is a covering array of strength t with an addituional property: Any set of d interactions each involving t parameters can be distinguished from any other such set by appearing in a distinct set of runs. If an array satisfies this condition it has the (d, t)-*locating property*.

More precisely, for array $A = (a_{ij})$ and t-way interaction T, define $\rho(A, T)$ as the set of runs (or, rows) of A in which T is covered. For a set $\mathscr{T}$ of t-way interactions, $\rho(A, \mathscr{T}) = \cup_{T \in \mathscr{T}} \rho(A, T)$.

Let $\mathscr{I}_t$ be the set of all t-way interactions for an array, and let $\overline{\mathscr{I}_t}$ be the set of all interactions of strength *at most* t. Consider a t-way interaction $T \in \overline{\mathscr{I}_t}$ of strength less than t. Any t-way interaction T' of strength t that contains T necessarily has $\rho(A, T') \subseteq \rho(A, T)$. A subset $\mathscr{T}'$ of t-way interactions in $\overline{\mathscr{I}_t}$ is *independent* if there do not exist $T, T' \in \mathscr{T}'$ with $T \subseteq T'$.

Definition 2.1 ((d, t)-Locating Array [7]) An array A is (d, t)-*locating* if whenever $\mathscr{T}_1, \mathscr{T}_2 \subseteq \overline{\mathscr{I}_t}$ and $\mathscr{T}_1 \cup \mathscr{T}_2$ is independent, $|\mathscr{T}_1| \leq d$, and $|\mathscr{T}_2| \leq d$, it holds that $\rho(A, \mathscr{T}_1) = \rho(A, \mathscr{T}_2) \Leftrightarrow \mathscr{T}_1 = \mathscr{T}_2$.

The covering array C in Table 1(a) does not have the $(1, 2)$-locating property because the set $\mathscr{T} = \{A_0B_1, A_0C_2, C_2D_1\}$ has $\rho(C, \mathscr{T}) = \{5\}$ for each $T \in \mathscr{T}$, i.e., each of the two-way interactions in each set of interactions in $\mathscr{T}$ appears only in run 5. However, the array L in Table 1(b) is $(1, 2)$-locating. Now, for each two-way interaction in $\mathscr{T}$ there is a row that distinguishes it from the others: $\rho(L, A_0B_1) = \{6, 7, 8, 9\}$, $\rho(L, A_0C_2) = \{5, 8, 9\}$, and $\rho(L, C_2D_1) = \{9, 13\}$.

Next we show how we use techniques from compressive sensing in analyzing locating arrays.

3 Analysis of Locating Arrays

3.1 The Screening Design and Model Matrix

A $(1, 2)$-locating array is proposed as the screening design for two reasons. The coverage and locating properties of such an array are essential for separating one- and two-way effects. As §3.2 describes, the proposed analysis method recovers the 'strongest' main effect or two-way interaction, one iteration at a time.

For the model matrix, a ± 1 compressive sensing matrix is proposed. A similar idea has been used for recovery of sparsifiable signals in communications and storage systems [2]. A *compressive sensing* (CS) matrix for an $N \times k$ $(1, \overline{2})$-locating array A has as many rows as A, and columns corresponding to the candidate terms [6]. Specifically, the CS matrix $M = (m_{ij})$ has N rows and $\sum_a \ell_a + \sum_{a \neq b} \ell_a \ell_b + 1$ columns where $m_{ij} = +1$ if effect j is covered in the ith run of A, and $m_{ij} = -1$ otherwise. A column of all $+1$ is required for the intercept. Table 2 shows the CS matrix for the locating array L in Table 1(b); for compactness of representation, we write ± instead of ±1 in the table and the column for the intercept is excluded. For example, the interaction BC can take on six values because B has two levels and C three; the third column of BC corresponds to the two-way interaction B_0C_2. Because this two-way interaction is present only in runs 5, 12, and 13 in the locating array, the column value in the compressive sensing matrix is set to +1 only in these rows. The reason for using a ±1 matrix instead of a binary matrix is to ensure that negating a column does not change the absolute value of its dot product with the residuals.

Table 2 LA with corresponding compressive sensing matrix for the running example

Locating Array		Compressive Sensing Matrix									
		A	B	C	D	AB	AC	AD	BC	BD	CD
		0 1	0 1	0 1 2	0 1 2	0 0 1 1	0 0 0 1 1 1	0 0 0 1 1 1	0 0 0 1 1 1	0 0 0 1 1 1	0 0 0 1 1 1 2 2 2
Run	A B C D					0 1 0 1	0 1 2 0 1 2	0 1 2 0 1 2	0 1 2 0 1 2	0 1 2 0 1 2	0 1 2 0 1 2 0 1 2
1	0 0 0 0	+ -	+ -	+ - -	+ - -	+ - - -	+ - - - - -	+ - - - - -	+ - - - - -	+ - - - - -	+ - - - - - - - -
2	0 0 0 1	+ -	+ -	+ - -	- + -	+ - - -	+ - - - - -	- + - - - -	+ - - - - -	- + - - - -	- + - - - - - - -
3	0 0 1 0	+ -	+ -	- + -	+ - -	+ - - -	- + - - - -	+ - - - - -	- + - - - -	+ - - - - -	- - - + - - - - -
4	0 0 1 1	+ -	+ -	- + -	- + -	+ - - -	- + - - - -	- + - - - -	- + - - - -	- + - - - -	- - - - + - - - -
5	0 0 2 2	+ -	+ -	- - +	- - +	+ - - -	- - + - - -	- - + - - -	- - + - - -	- - + - - -	- - - - - - - - +
6	0 1 0 2	+ -	- +	+ - -	- - +	- + - -	+ - - - - -	- - + - - -	- - - + - -	- - - - - +	- - + - - - - - -
7	0 1 1 2	+ -	- +	- + -	- - +	- + - -	- + - - - -	- - + - - -	- - - - + -	- - - - - +	- - - - - + - - -
8	0 1 2 0	+ -	- +	- - +	+ - -	- + - -	- - + - - -	+ - - - - -	- - - - - +	- - - + - -	- - - - - - + - -
9	0 1 2 1	+ -	- +	- - +	- + -	- + - -	- - + - - -	- + - - - -	- - - - - +	- - - - + -	- - - - - - - + -
10	1 0 0 2	- +	+ -	+ - -	- - +	- - + -	- - - + - -	- - - - - +	+ - - - - -	- - + - - -	- - + - - - - - -
11	1 0 1 2	- +	+ -	- + -	- - +	- - + -	- - - - + -	- - - - - +	- + - - - -	- - + - - -	- - - - - + - - -
12	1 0 2 0	- +	+ -	- - +	+ - -	- - + -	- - - - - +	- - - + - -	- - + - - -	+ - - - - -	- - - - - - + - -
13	1 0 2 1	- +	+ -	- - +	- + -	- - + -	- - - - - +	- - - - + -	- - + - - -	- + - - - -	- - - - - - - + -
14	1 1 0 0	- +	- +	+ - -	+ - -	- - - +	- - - + - -	- - - + - -	- - - + - -	- - - + - -	+ - - - - - - - -
15	1 1 0 1	- +	- +	+ - -	- + -	- - - +	- - - + - -	- - - - + -	- - - + - -	- - - - + -	- + - - - - - - -
16	1 1 1 0	- +	- +	- + -	+ - -	- - - +	- - - - + -	- - - + - -	- - - - + -	- - - + - -	- - - + - - - - -
17	1 1 1 1	- +	- +	- + -	- + -	- - - +	- - - - + -	- - - - + -	- - - - + -	- - - - + -	- - - - + - - - -
18	1 1 2 2	- +	- +	- - +	- - +	- - - +	- - - - - +	- - - - - +	- - - - - +	- - - - - +	- - - - - - - - +

3.2 The Screening Method

To achieve a small run-size, locating arrays often exhibit highly unbalanced structure. This requires the development of a method for screening that can cope with imbalance.

The proposed screening method has two steps. First, a *breadth-first search* (BFS) algorithm is developed to identify a user-specified number of models that are the 'best' explanations of a response using *orthogonal matching pursuit* (OMP) [38]. Secondly, using the models produced in the BFS search, the *screening algorithm* aggregates main effects and two-way interactions to identify the candidate important effects and factors. The 'many-model' method [15] also retains a fraction of best models based on error sum of squares, but their method does not appear to scale to large numbers of factors.

3.2.1 The Breadth-First Search (BFS) Algorithm

The BFS algorithm is parameterized by three user-specified variables: n_{models} giving the number of fitted models that the algorithm returns, n_{new} giving the fan-out (i.e., number of children) of each node in the BFS tree, and n_{terms} giving the number of effects in each of the final fitted models. In the BFS tree, the nodes at level ℓ correspond to fitted models with ℓ effects. The algorithm generates a tree of height n_{terms}.

The BFS algorithm is given in Algorithm 1. The root of the tree is a single model consisting of the mean response and a score initialized to zero. A BFS expands each node at level ℓ to n_{new} nodes at level $\ell + 1$ (line 8). For efficiency, the search tree is stored implicitly, with each model of length ℓ stored in priority queue q_ℓ; we retain only the top fifty models ordered by R^2, the proportion of the variability in the data explained by the mode [27]. Each child expands the fitted model of its parent by adding the ith most important effect for $1 \leq i \leq n_{new}$ using OMP (line 9). Specifically, the ith most important effect corresponds to the ith effect in the ranking of the absolute values of the dot products or correlations of each column in the CS matrix with the current residual vector. Ordinary least squares (OLS) [31] is used to update coefficient estimates, after which the residuals and score of the added effect are updated (lines 10-14). (OMP for logistic regression [23] can be used if the response measured in experimentation is binary.)

The increment in R^2 of the expanded model that results from adding the ith effect is used as the score of the effect. The model, its residuals, its R^2 and adjusted R^2 (a variation of the the ordinary R^2 statistic that reflects the number of terms in the model [27]), and its scores are then inserted into the priority queue of length $\ell + 1$ (line 15). These steps are repeated until a stopping criterion is met. To simplify the algorithm it stops when each model has n_{terms} effects (line 5). The model matrix, stopping criterion, and scoring method of the BFS algorithm could each be chosen differently.

When only main effects and two-way interactions are considered, a $(1, 2)$-locating array suffices. All effects are separable under such a design, i.e., all columns in the compressive sensing matrix are different. For binary factors, the absolute values for the two main-effects columns are equal, and either can be selected. The dot product is easy to compute, and ranks effects based on absolute correlation with residuals of the current model.

It is possible for duplicate fitted models to arise in q_ℓ, e.g., when the same terms are selected but in a different order. While only unique fitted models are kept in the queue, duplicates are accounted for by adding scores of each effect from the duplicate. Thus duplication is not ignored, and more distinct models are explored.

3.2.2 The Screening Algorithm

In screening, the interest is in identifying a few important main effects and two-way interactions. One approach is to examine the scores of the effects in the list of

Algorithm 1 BFS(*effects*, M, *data*, n_{models}, n_{terms}, n_{new})

Input: List of candidate *effects*, compressive sensing matrix M, response vector $data$, number of fitted models n_{models} to return, number of effects in each final fitted model n_{terms}, fan-out of the BFS tree n_{new}

Output: List of n_{models} best fitted models ranked by R^2 with n_{terms} terms each

$model_{new} \leftarrow$ mean of data
$residuals_{new} \leftarrow data - model_{new}$
$scores_{new} \leftarrow 0$
enqueue(q_1, ($model_{new}$, $residuals_{new}$, R^2, $adjR^2$, $scores_{new}$))
for $\ell \leftarrow 1, \ldots, n_{terms}$ **do**
  while q_ℓ has models **do**
    ($model$, $residuals$, R^2, $adjR^2$, $scores$) $\leftarrow dequeue(q_\ell)$
    for $i \leftarrow 1, \ldots, n_{new}$ **do**
      $effects_k \leftarrow \arg\max_i |M_i \cdot residuals|$
      $model_{new} \leftarrow$ OLS($effects(model) \cup effects_k$, $data$)
      $residuals_{new} \leftarrow data - model_{new}$
      $R^2{}_{new} \leftarrow R^2$ value of $model_{new}$
      $adjR^2{}_{new} \leftarrow$ adjusted R^2 value of $model_{new}$
      $scores_{new} \leftarrow$ append increment in R^2 attributed to $effects_k$ to $scores$
      enqueue($q_{\ell+1}$, ($model_{new}$, $residuals_{new}$, $R^2{}_{new}$, $adjR^2{}_{new}$, $scores_{new}$))
    end for
  end while
end for
return list of n_{models} fitted models from $q_{n_{terms}}$ ranked by R^2 value

fitted models produced, and select those with higher scores. Instead, the approach in Algorithm 2 aggregates over effects of all n_{models} models without explicitly considering levels.

Algorithm 2 Screening(*effects*, $q_{n_{terms}}$, n_{models}, n_{terms})

Input: List of all candidate *effects*, the list of fitted models $q_{n_{terms}}$ and corresponding scores from Algorithm 1, the number of fitted models n_{models} in the list, the number of effects n_{terms} in each fitted model

Output: A list of effects in non-increasing order by score

Initialize the score of each effect to zero
Store the n_{models} and their corresponding scores in a priority queue q
for $i \leftarrow 1, \ldots, n_{models}$ **do**
  $(model_i, scores_i) \leftarrow dequeue(q)$
  for $j \leftarrow 1, \ldots, n_{terms}$ **do**
    $k \leftarrow$ index of *effects* corresponding to term j in $model_{i_j}$
    $effect\text{-}score_k = effect\text{-}score_k + scores_{i_j}$
  end for
end for
return list of *effects* ranked by *effect-score*

Effects are reported in non-increasing order by their aggregate scores to support user-interpretation of the results. Screening results are usually reported considering heredity, therefore if an interaction effect $X \times Y$ is reported as active, then X and Y are also considered active [22].

4 Experimental Set-up

4.1 The `w-iLab.t` Testbed

`w-iLab.t` is an advanced testbed that is used to perform heterogeneous wireless experimentation [3]. It is pseudo-shielded from external interference and is equipped with various wireless technologies, including IEEE 802.11, IEEE 802.15.4, Bluetooth dongles, Software Defined Radios (SDRs), LTE femto cells, and others. The `w-iLab.t` testbed is part of Emulab and uses the cOntrol Management Framework (OMF) for resource allocation, hardware and software configuration, and the orchestration of experiments. Finally, measurement data from each run is collected over a wired control network and stored in a central database for further processing.

4.2 Wi-Fi Conferencing Scenario

As a representative use case, a large-scale wireless conferencing scenario is considered. A high-level representation of the Wi-Fi conferencing scenario created in the `w-iLab.t` testbed is shown in Fig. 1. It is composed of a speaker node broadcasting voice traffic over a Wi-Fi network and listener nodes receiving and playing the transmitted packets. The speaker can configure 22 different parameters (described in §4.3) that may influence the transmissions. The listeners continuously calculate audio quality and RF transmission exposure.

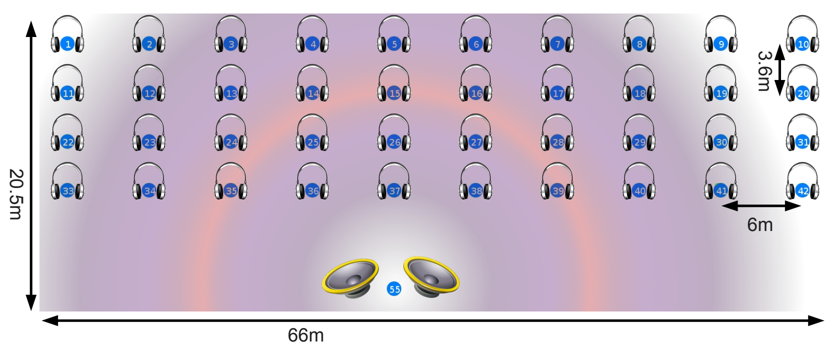

Fig. 1 The Wi-Fi conference scenario as mapped to the wireless testbed. Listener nodes are in the first 4 rows and the speaker node is positioned at the bottom center.

To orchestrate the experiment, an OMF script processes the experimental design given by the LA and iteratively executes each run. For the execution of each run, the system is first brought to a known state by resetting all wireless interfaces and caches of each node, followed by configuration of the parameters as specified by the run. After a warm-up period to avoid transient effects, measurements are collected. Table 3 shows the list of resources used for the Wi-Fi conferencing scenario.

Table 3 Experiment resource description

Resource	Description
Wi-Fi chipset	Atheros Sparklan WPEA-110N/E/11n
Wi-Fi driver	ath9k
OS	Ubuntu 14.04 LTS
kernel	Linux 3.13.0-33-generic

4.3 Selected Parameters and Values

The testbed nodes can be configured by uploading an image containing the operating system and application to run. We selected 22 parameters from the kernel's IP and UDP protocols, the Wi-Fi card driver, the audio codec used in our application, and a source of interference implemented via a dedicated SDR. Each parameter has from 2 to 5 values. Categorical parameters included all settings as levels, while numerical (i.e., continuous) parameters had their levels spaced exponentially to avoid giving preference to a particular scale. For each parameter, we also ensure that the default assigned by the Linux kernel and/or user-space tools is present. The full list of parameters and values is provided in Table 4.

We also used transmission power assignments 6 dBm lower for 2.4 GHz than for 5 GHz so that propagation effects would be approximately equal for the different bands, due to the *free-space path loss* difference between these frequency bands [24].

4.4 Performance Metrics

Two performance metrics we measured during experimentation: audio quality and radio frequency (RF) exposure. The audio quality is quantified using an aggregate *mean opinion score* (MOS) [26] over the complete audio transmit path. The audio quality is first affected by the encoding process at the transmitter side and further reduced when transmitted over the air (see Fig. 2).

Within the encoder unit, the first quality loss is introduced as a function of the encoder bitrate, type of encoder, and audio class used. Afterwards, the audio is

Table 4 Parameters and values used in the scenario (default values in bold)

Parameter	Identifier	Values
Band	`band`	2.4, 5 GHz
Channel	`channel`	1, 6, 11 (in 2.4 GHz band); 36, 40, 44 (in 5 GHz band)
Wi-Fi bitrate	`bitrate`	6, 9, 12, 24, 36 Mbps
Transmit power	`txpower`	1, 2, 4, 7, 10 dBm (2.4 GHz); 7, 8, 10, 13, **16** dBm (5 GHz)
MTU	`mtu`	256, 512, 1024, 1280, **1500** bytes
Transmit queue length	`txqueuelen`	10, 50, 100, 500, **1000** packets
Queuing discipline	`qdisc`	pfifo, bfifo, **pfifo_fast**
IP fragment low threshold	`ipfrag_low_thresh`	25%, 50%, **75%**, 100% of high threshold
IP fragment high threshold	`ipfrag_high_thresh`	16384, 65536, 262144, 1048576, **4194304** bytes
UDP receive buffer minimum	`udp_rmem_min`	**1.9231%**, 10%, 50% of maximum
UDP receive buffer default	`rmem_default`	0%, 25%, 50%, 75%, **100%** from minimum to maximum
UDP receive buffer maximum	`rmem_max`	2304, 10418, 47105, **212992** bytes
UDP transmit buffer minimum	`udp_wmem_min`	**1.9231%**, 10%, 50% of maximum
UDP transmit buffer default	`wmem_default`	0%, 25%, 50%, 75%, **100%** from minimum to maximum
UDP transmit buffer maximum	`wmem_max`	4608, 16537, 59349, **212992** bytes
UDP global buffer minimum	`udp_mem_min`	25%, **50%**, 75% of maximum
UDP global buffer pressure	`udp_mem_pressure`	0%, **33.338%**, 50%, 75%, 100% from minimum to maximum
UDP global buffer maximum	`udp_mem_max`	95, 949, 9490, **94896** pages
Audio codec	`codec`	Opus, Speex
Audio codec bitrate	`codecBitrate`	7600, 16800, 24000, 34000 bit/s (or nearest allowed by codec)
Frame length aggregation	`frameLen`	20, 40, 60
Interference channel occupancy	`intCOR`	10%, 25%, 50%, 75%, 90%

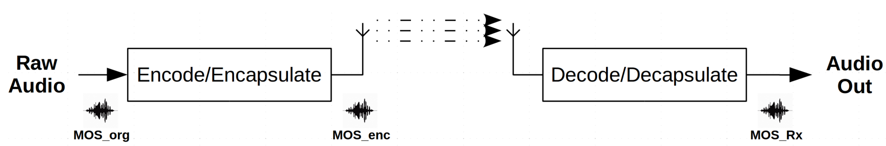

Fig. 2 The audio quality degradation is calculated in two phases: once after the encoder unit and later after the wireless transmission.

transmitted over the air and a second quality loss is introduced due to packet loss, jitter, and latency impairments.

Radio frequency (RF) transmission exposure calculates the electromagnetic energy absorbed by a human body due to uplink and downlink wireless transmissions [25]. The RF *exposure index* (EI) is measured in *specific absorption ratio* (SAR) units of a given amount of power (Watts) over a given mass of human body (kg). The formula for exposure by Varsier et al. [39], generalized in Mehari [24], covers a wide range of categories (i.e., population, environment, radio access technologies, load profile, posture) but specific to our scenario, the formula used is

$$EI^{SAR} = \frac{1}{T}\left[\sum_{t}^{N_T}\left(d^{UL}\overline{P}_{TX}\right) + d^{DL}\overline{S}_{\text{inc}}\right].$$

During a given time frame T, where N_T is the number of periods within the time frame, a transmitting antenna induces an exposure to a speaker proportional to the transmitted power $\overline{P}_{TX}$ and also induces an exposure to far away listeners that is proportional to the incident power density $\overline{S}_{\text{inc}}$. After that, the electromagnetic energy absorption per kilogram of body mass is calculated by applying the uplink and downlink absorption parameters $d^{UL} = 0.0070$ W/kg for 1W of transmitted power and $d^{DL} = 0.0028$ W/kg for $1W/m^2$ of received power density respectively [29].

5 Results

Seidel et al. [33] discuss two randomized algorithms for constructing locating arrays based on the Stein-Lovász-Johnson paradigm, and the Lovász Local Lemma. The implementation of these algorithms, as well as the implementation of the analysis method is publicly available [32]; these were used to generate the LA and analyze the results collected that are described here.

Using the proposed analysis method based on a 73-run $(1, 2)$-locating array, Table 5(a) shows the top five effects and their scores for the audio quality when $n_{terms} = 9$; in this case, the fitted models have $0.74 \le R^2 \le 0.76$. For $n_{terms} = 21$, all the fitted models have $R^2 \ge 0.96$ and the top five effects and their scores are shown in Table 5(b). The proposed screening method identifies `txpower` and the interaction `intCOR` × `band` as having a significant impact on the response of audio quality.

Table 5(c) lists the top five effects and their scores for RF exposure when $n_{terms} = 10$; these fitted models have $R^2 \ge 0.96$. The proposed screening method identifies `txpower` and `band` as the important factors for the RF exposure.

Table 5 Top five terms and their score for audio quality when (a) $n_{terms} = 9$; (b) $n_{terms} = 21$; and for (c) RF exposure when $n_{terms} = 10$, in the Wi-Fi conferencing experiment

(a)

Term	Score
intCOR×band	325.51
txpower	320.00
intCOR	68.29

(b)

Term	Score
txpower	24961.10
intCOR×band	18237.70
intCOR	14748.40

(c)

Term	Score
txpower	1489.92
band	1035.76
bitrate	589.91

We also analyze the data from the wireless network experiment using two additional methods: The Dantzig selector method [28] and Lasso regression [37]. For each method, we use two different coding schemes: Dummy coding and orthogonal polynomial coding.

For the Dantzig selector we use the R package `flare` [20]. We scale the columns of the model matrix, and set the minimum value for the tuning parameter $\delta_{Dantzig} = 0.1$ and the number of $\delta_{Dantzig}$ to 11; see [28] for how to choose these parameters. We fix γ (a threshold between signal and noise) at zero so that we can keep all the selected effects. In Lasso regression, we use the R package `glmnet` for analysis [10, 11].

Table 6 summarizes the terms found by the proposed screening method, the Dantzig selector method, and Lasso regression. There is good agreement on the screening results for both responses when the polynomial model is used. However with dummy-coding, neither the Dantzig selector nor the Lasso regression method appears to be as accurate.

Table 6 Screening results for the Wi-Fi conferencing experiment listed by the method used

Method	**Audio Quality**	**RF Exposure**
Proposed method	txpower intCOR band	txpower band bitrate
Dantzig selector (polynomial coding)	txpower intCOR band	txpower band bitrate
Dantzig selector (dummy coding)	txpower band rate udp_mem_pressure ipfrag_high_thresh	band txpower
Lasso regression (polynomial coding)	txpower intCOR band	txpower band rate
Lasso regression (dummy coding)	band bitrate txpower	txpower band

6 Conclusions and Future Work

In this paper, a locating array is used to screen parameters and two-way interactions affecting MOS and RF exposure in a Wi-Fi conferencing experiment run in `w-iLab.t`, a complex engineered wireless network. It, together with our compressive sensing based analysis method, is able to screen out most of the parameters and two-way interactions as insignificant. The results are validated by the Dantzig selector and Lasso regression methods.

Because our analysis method can fail to yield a good model if the residuals end up farther from the term that should be selected than from some other term, it may be possible to produce a locating array that has better analysis properties by guiding the choices so as to maximize the minimum distance between any pair of terms' associated vectors. An investigation of alternate stopping criteria is also of interest.

An additional direction for future extension is to handle parameters that can be measured but not controlled, such as temperature or background interference, and potentially even to detect indirectly the presence of parameters that have a significant effect but cannot be directly measured.

Acknowledgements We appreciate the useful comments on our paper made by the reviewers. Thanks to Stephen Seidel, Brandon Hang, Randy Compton, Yasmeen Akhtar, and John Stufken over the years for their contributions to the work on this project.

This work is supported in part by the U.S. National Science Foundation under Grant No. 1421058. This work is also supported by an IWT project, and by the European Commission Horizon 2020 Programme under grant agreement n 688116 (eWINE).

References

1. Athanasiou, G., Korakis, T., Ercetin, O., Tassiulas, L.: A cross-layer framework for association control in wireless mesh networks. IEEE Transactions on Mobile Computing **8**(1), 65–80 (2009)
2. Baraniuk, R.G.: Compressive sensing. IEEE Signal Processing Magazine **24**(4), 118–121 (2007)
3. Bouckaert, S., Vandenberghe, W., Jooris, B., Moerman, I., Demeester, P.: The w-iLab.t testbed. In: T. Magedanz, A. Gavras, N.H. Thanh, J.S. Chase (eds.) Testbeds and Research Infrastructures. Development of Networks and Communities, pp. 145–154. Springer Berlin Heidelberg (2011). DOI 10.1007/978-3-642-17851-1_11
4. Cáceres, R., Iftode, L.: Improving the performance of reliable transport protocols in mobile computing environments. IEEE Journal on Selected Areas in Communications **13**(5), 850–857 (1995)
5. Cammarano, A., Presti, F.L., Maselli, G., Pescosolido, L., Petrioli, C.: Throughput-optimal cross-layer design for cognitive radio ad hoc networks. IEEE Transactions on Parallel and Distributed Systems **26**(9), 2599–2609 (2015)
6. Candès, E., Wakin, M.: An introduction to compressive sampling. IEEE Signal Processing Magazine **25**(2), 21–30 (2008)
7. Colbourn, C.J., McClary, D.W.: Locating and detecting arrays for interaction faults. Journal of Combinatorial Optimization **15**(1), 17–48 (2008)
8. Croarkin, C., Tobias, P., Filliben, J.J., Hembree, B., Guthrie, W., Trutna, L., Prins, J. (eds.): NIST/SEMATECH e-Handbook of Statistical Methods. NIST (2012)
9. Efazati, S., Azmi, P.: Cross layer power allocation for selection relaying and incremental relaying protocols over single relay networks. IEEE Transactions on Wireless Communications **15**(7), 4598–4610 (2016)
10. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software **33**(1), 1–22 (2010). URL `https://www.jstatsoft.org/v33/i01/`

11. Friedman, J., Hastie, T., Tibshirani, R., Narasimhan, B., Tay, K., Simon, N.: Lasso and elastic-net regularized generalized linear models, R package glmnet. URL `https://glmnet.stanford.edu/`
12. Gilmour, S.G.: Factor screening via supersaturated designs. In: A.M. Dean, S.M. Lewis (eds.) Screening: Methods for Experimentation in Industry, Drug Discovery and Genetics, chap. 8, pp. 169–190. Springer-Verlag (2006)
13. Hang, B.: Wireless conferencing experiments using the bash experiment orchestration framework. URL `https://github.com/bmhang/wireless-conference-guide`
14. Hartman, A.: Software and hardware testing using combinatorial covering suites. In: M.C. Golumbic, I.B.A. Hartman (eds.) Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications, chap. 10, pp. 237–266. Springer US, Boston, MA (2005). DOI 10.1007/0-387-25036-0_10
15. Holcomb, D.R., Montgomery, D.C., Carlyle, W.M.: Analysis of supersaturated designs. Journal of Quality Technology **35**(1), 13–27 (2003). DOI 10.1080/00224065.2003.11980188
16. Hu, W., Yousefi'zadeh, H., Li, X.: Load adaptive MAC: A hybrid MAC protocol for MIMO SDR MANETs. IEEE Transactions on Wireless Communications **10**(11), 3924–3933 (2011)
17. Jones, B., Majumdar, D.: Optimal supersaturated designs. Journal of the American Statistical Association **109**(508), 1592–1600 (2014)
18. Kleijnen, J.P.C.: An overview of the design and analysis of simulation experiments for sensitivity analysis. European Journal of Operational Research **164**, 287–300 (2005)
19. Li, R., Lin, D.K.J.: Analysis methods for supersaturated designs: Some comparisons. Journal of Data Science pp. 249–260 (2003)
20. Li, X.: Family of Lasso regression, R package flare (2013). URL `https://www.rdocumentation.org/packages/flare/versions/0.9.8/topics/flare-package`
21. Li, X., Sudarsanam, N., Frey, D.D.: Regularities in data from factorial experiments. Complexity **11**(5), 32–45 (2006). DOI 10.1002/cplx.20123
22. Li, X., Sudarsanam, N., Frey, D.D.: Regularities in data from factorial experiments. Complexity **11**(5), 32–45 (2006). DOI http://dx.doi.org/10.1002/cplx.20123
23. Lozano, A.C., Świrszcz, G., Abe, N.: Group orthogonal matching pursuit for logistic regression. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), vol. PMLR 15, pp. 452–460 (2011)
24. Mehari, M.T.: Performance optimization and modelling of complex wireless networks using surrogate models. Ph.D. thesis, Ghent University (2018)
25. Mehari, M.T., De Poorter, E., Couckuyt, I., Deschrijver, D., Gerwen, J.V.V., Pareit, D., Dhaene, T., Moerman, I.: Efficient global optimization of multi-parameter network problems on wireless testbeds. Ad Hoc Networks **29**, 15–31 (2015). DOI http://dx.doi.org/10.1016/j.adhoc.2015.01.014
26. Mehari, M.T., Poorter, E.D., Couckuyt, I., Deschrijver, D., Vermeeren, G., Plets, D., Joseph, W., Martens, L., Dhaene, T., Moerman, I.: Efficient identification of a multi-objective Pareto front on a wireless experimentation facility. IEEE Transactions on Wireless Communications **15**(10), 6662–6675 (2016). DOI 10.1109/TWC.2016.2587261
27. Montgomery, D.C.: Design and Analysis of Experiments, 9th edn. John Wiley and Sons, Inc. (2017)
28. Phoa, F.K.H., Pan, Y.H., Xu, H.: Analysis of supersaturated designs via the Dantzig selector. J. Statist. Plann. Inference **139**(7), 2362–2372 (2009). DOI 10.1016/j.jspi.2008.10.023
29. Plets, D., Joseph, W., Vanhecke, K., Vermeeren, G., Wiart, J., Aerts, S., Varsier, N., Martens, L.: Joint minimization of uplink and downlink whole-body exposure dose in indoor wireless networks. Biomed Research International p. 9 (2015)
30. SAS Institute Inc.: JMP®, Version 16 (1989–2022). URL `https://www.jmp.com/`
31. Searle, S.R.: Linear models for unbalanced data. John Wiley & Sons (1987)
32. Seidel, S.A.: Tools for locating arrays (2019). URL `https://github.com/sseidel16/v4-la-tools`
33. Seidel, S.A., Sarkar, K., Colbourn, C.J., Syrotiuk, V.R.: Separating interaction effects using locating and detecting arrays. In: C. Iliopoulos, H.W. Leong, W.K. Sung (eds.) Combinatorial Algorithms, pp. 349–360. Springer International Publishing, Cham (2018)

34. Shakkottai, S., Rappaport, T.S., Karlsson, P.C.: Cross-layer design for wireless networks. IEEE Communications Magazine **41**(10), 74–80 (2003)
35. Stinson, D.R.: Combinatorial Designs: Constructions and Analysis. Springer-Verlag New York, Inc. (2004)
36. Syrotiuk, V.R., Colbourn, C.J.: Meta-modelling for complex engineered networks, U.S. National Science Foundation award 1421058. URL `http://www.public.asu.edu/~syrotiuk/meta-modelling.html`
37. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996). DOI 10.1111/j.2517-6161.1996.tb02080.x
38. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans. Inform. Theory **53**(12), 4655–4666 (2007). DOI 10.1109/TIT.2007.909108. URL `https://doi-org.ezproxy1.lib.asu.edu/10.1109/TIT.2007.909108`
39. Varsier, N., Plets, D., Corre, Y., Vermeeren, G., Joseph, W., Aerts, S., Martens, L., Wiart, J.: A novel method to assess human population exposure induced by a wireless cellular network. Bio Electro Magnetics **36**(6), 451–463 (2015)
40. Wang, Y., Vuran, M.C., Goddard, S.: Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks. IEEE/ACM Transactions on Networking **20**(1), 305–318 (2012)