Locating Arrays: A New Experimental Design for Screening Complex Engineered Systems

Abraham N. Aldaco, Charles J. Colbourn, and Violet R. Syrotiuk
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, U.S.A. 85287-8809
{aaldacog, colbourn, syrotiuk}@asu.edu

ABSTRACT

The purpose of a screening experiment is to identify significant factors and interactions on a response for a system. Engineered systems are complex in part due to their size. To apply traditional experimental designs for screening in complex engineered systems requires either restricting the factors considered, which automatically restricts the interactions to those in the set, or restricting interest to main effects, which fails to consider any possible interactions. To address this problem we propose a *locating array* (LA) as a screening design. Locating arrays exhibit logarithmic growth in the number of factors because their focus is on identification rather than on measurement. This makes practical the consideration of an order of magnitude more factors in experimentation than traditional screening designs. We present preliminary results applying an LA for screening the response of TCP throughput in a simulation model of a mobile wireless network. The full-factorial design for this system is infeasible (over 10^{43} design points!) yet an LA has only 421 design points. We validate the significance of the identified factors and interactions independently using the statistical software JMP. Screening using locating arrays is viable and yields useful models.

Categories and Subject Descriptors

General and reference [**Cross-computing tools and techniques**]: Experimentation; Mathematics of computing [**Discrete mathematics**]: Combinatorics

General Terms

Experimentation

Keywords

Screening experiments, Locating arrays

1. INTRODUCTION

Computer and networked systems are examples of *complex engineered systems* (CESs). The complexity of an engineered system is not just due to its size, but also arises from its structure, operation (including control and management), evolution over time, and that people are involved in its design and operation [35].

Experimentation is often used to study the performance of CESs. At its most basic, a system may be viewed as transforming some input variables, or *factors*, into one or more observable output variables, or *responses*. Some factors of a system are *controllable*, whereas others are not.

Objectives of experimentation include:

Screening: Which factors and interactions are most influential on a response?

Confirmation: Is the system currently performing in the same way as it did in the past?

Discovery: What happens when new operating conditions, materials, factors, *etc*., are explored?

Robustness: Under what conditions does a response degrade?

Stability: How can variability in a response be reduced?

Our focus is on screening using techniques from statistical *design of experiments* (DoE). DoE refers to the process of planning an experiment so that appropriate data are collected and analyzed by statistical methods, in order to result in valid and objective conclusions. Hence any experimental problem includes both the design of the experiment and the statistical analysis of the data.

Suppose that there are k factors, $F_1, \dots, F_k$, and that each factor F_j has a set $L_j = \{v_{j,1}, \dots, v_{j,\ell_j}\}$, of ℓ_j possible *levels* (or values). A *design point* is an assignment of a level from L_j to F_j, for each factor $j = 1, \dots, k$. An *experimental design* is a collection of design points. When a design has N design points, it can be represented by an $N \times k$ array $A = (a_{i,j})$ in which each row i corresponds to a design point and each column j to a factor; the entry $a_{i,j}$ gives the level assigned to factor j in the ith design point. When run, a design point results in one or more observable responses.

A *t-way interaction* (or interaction of *strength* t) in A is a choice of t columns $i_1, \dots, i_t$, and the selection of a level $\nu_{i_j} \in L_{i_j}$ for $1 \leq j \leq t$, represented as $T = \{(i_j, \nu_{i_j}) : 1 \leq j \leq t\}$. Every design point in A *covers* $\binom{k}{t}$ interactions of strength t.

When the objective of experimentation is screening, it is often recommended to keep the number of factors low. It has been considered impractical to experiment with "many" factors; about ten factors is a suggested maximum [23, 31]. Generally, two levels for each factor is considered to work well in screening experiments.

Methods for screening seek to reduce the number of design points required because the exhaustive *full-factorial design* [9, 31] is too large. For k factors each with two levels it has 2^k design points. An *analysis of variance* (ANOVA) allows the significant factors and interactions on the response to be identified.

A *fractional factorial* design 2_R^{k-p} is a $\frac{1}{2^p}$ fraction of a full factorial design with k two-level factors. The design is described by p

generators, expressions of factors that are confounded; the generators determine the alias structure. A design is of *resolution* R if no m-factor effect is aliased with another effect containing fewer than $R - m$ factors.

A *D-optimal* design is a popular experimental design among those using optimality criteria. A model to fit, and a bound N on the number of design points, must be specified *a priori*; this restricts the factors to be analyzed to those in the model. The size of a D-optimal design is bounded by the size of a full-factorial design.

Some designs aggregate the factors into groups, *e.g.*, sequential bifurcation [24], to improve design efficiency. Grouping requires care to ensure that factor effects do not cancel. This presents a "chicken and egg" problem: we need to know how to group in order to group. Often, a domain expert is expected to make such grouping decisions. While such experts may have considerable knowledge, it is doubtful whether an expert knows the importance of a specific factor or interaction in a CES.

An *interaction graph* depicts how a change in the level of one factor affects the other factor with respect to a response. Figure 1 shows an interaction graph for the factors of routing and *medium access control* (MAC) protocol on average delay in a network. The choice of MAC protocol (EDCF or IEEE 802.11) has little impact on the average delay in the AODV routing protocol, while for the DSR routing protocol the impact is very large; see [53]. If MAC protocols were aggregated, this significant interaction would be lost.

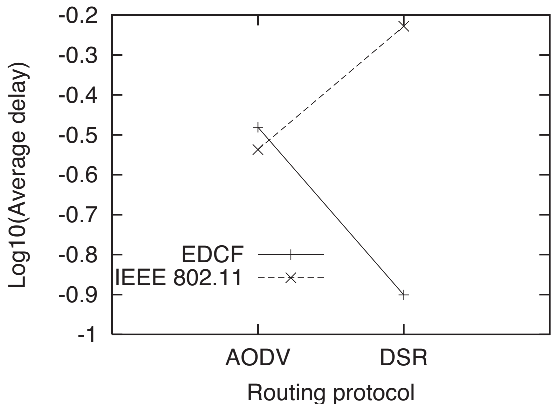


Figure 1: Interaction of routing and MAC protocols on delay [53].

A fractional factorial design is *saturated* when it investigates $k = N - 1$ factors in N design points [31]. In a *supersaturated design*, the number of factors $k > N - 1$; such designs contain more factors than design points. These designs are only able to estimate a main effects model [27, 31]. Thus they cannot consider possible interactions at all.

Even with substantial and detailed domain knowledge, it is imperative not to eliminate or aggregate factors *a priori*. Our goal, therefore, is an automatic and objective approach to screening. To address this problem we have formulated the definition of a *locating array* (LA) [8]. Locating arrays exhibit logarithmic growth in the number of factors because their focus is on identification rather than on measurement. This makes practical the consideration of an order of magnitude more factors in experimentation, removing the need for the elimination of factors. As a result, LAs have the potential to transform experimentation in huge factor spaces such as those found in CESs.

The rest of this paper is organized as follows. §2 defines a locating array, and gives an example of how a design is used for location. §3 presents preliminary results applying an LA for screening the response of TCP throughput in a simulation model of a mobile wireless network. The full-factorial design for this system is infeasible — it has over 10^{43} design points! Yet there is an LA with only 421 design points. We develop an algorithm using the LA to identify the significant factors and interactions from the data collected, providing a small example. In §4 we validate the significance of the identified factors and interactions independently using the statistical software JMP. Finally, in §5 we summarize, discuss potential threats to our approach, directions for this research, and conclude.

2. LOCATING ARRAYS

Reducing the number of design points required relies on a *sparsity of effects* assumption, that interactions of interest involve at most a small, known number t of interacting factors. As one means of reduction, we define *locating arrays* (LAs) [8]. For a set of factors each taking on a number of levels, an LA permits the identification of a small number of significant interactions among small sets of (factor, level) combinations.

LAs differ from standard designed experiments, which are used to *measure* interactions and to develop a model for the response as a function of these [31]. "Search designs" [17, 48, 49] also attempt to locate interactions of higher strength, but their focus remains on measurement and hence on balanced designs. Rao [20] shows that the number of design points in a balanced design must be at least as large as the number of interactions considered. Thus if t-way interactions among k factors each having v levels are to be examined, balanced designs only reduce the v^k exhaustive design points to $O(k^t)$. The selection of few factors from hundreds of candidates by this reduction is not viable. By lessening the requirement from measurement to identification, LAs are not subject to the Rao bound.

Fortunately LAs behave more like *covering arrays*, experimental designs in which every t-way interaction among factors appears in at least one design point. Unlike designed experiments, the number of design points in a covering array for k factors grows as a *logarithmic* function of k (see [43], for example). In [8], a construction of LAs using covering arrays of higher strength is given, and hence LAs also exhibit this logarithmic growth, making them asymptotically much more efficient than balanced designs. This motivates the consideration of covering arrays, which have been the subject of extensive study [4, 5, 19, 36]. They are used in testing software [10, 13, 25, 26], hardware [46, 50], composite materials [3], biological networks [44, 47], and others. Their use to facilitate location of interactions is examined in [29, 56], and measurement in [21, 22]. Covering arrays form the basis for combinatorial methods to learn an unknown classification function using few evaluations — these arise in computational learning and classification, and hinge on locating the relevant attributes (factors) [11]. Algorithms for generating covering arrays range from greedy (*e.g.*, [2, 16]) through heuristic search (*e.g.*, [38, 52]). However, combinatorial constructions (see [5]) provide the only available deterministic means of producing covering arrays with more than a few hundred factors.

A design point, when run, yields one or more responses. For ease of exposition, we classify the responses in two groups, those that exceed a specified threshold and those that do not. So we suppose that the outcome of a run of a design point is a single binary response ("pass" or "fail"). A *fault* is caused by one or more t-way interactions, and is evidenced by a run failing.

Given an experimental design and the set of interactions that cause faults, the outcomes can be easily calculated: A run fails exactly when it contains one or more of the faulty interactions, and does not fail otherwise. In order to observe a fault, the interaction must be covered by at least one design point. With no restriction on the interactions that can cause faults, every interaction must be covered. Then the best one can do is to form all $\prod_{j=1}^{k} \ell_j$ possible design points, the exhaustive design. Using sparsity of effects, an upper bound t is placed on the strength of interactions that may be faulty. Then we require that every t-way interaction be covered; in other words, the design is a covering array of strength t.

Let $A = (a_{i,j})$ be an experimental design, an $N \times k$ array where in each row i, levels in the jth column are chosen from a set L_j of size ℓ_j. For array A and t-way interaction $T = \{(i_j, \nu_{i_j}) : 1 \leq j \leq t\}$, define $\rho(A,T) = \{r : a_{r,i_j} = \nu_{i_j}, 1 \leq j \leq t\}$ as the set of rows of A in which T is covered. For a set $\mathscr{T}$ of interactions, $\rho(A,\mathscr{T}) = \cup_{T\in\mathscr{T}} \rho(A,T)$. Locating faults requires that $\mathscr{T}$ be recovered from $\rho(A,\mathscr{T})$, whenever $\mathscr{T}$ is a possible set of faults.

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 an interaction $T \in \overline{\mathscr{I}_t}$ of strength less than t. Any interaction T' of strength t that contains T necessarily has $\rho(A,T') \subseteq \rho(A,T)$. In this case, when T is faulty we are unable to determine whether or not T' is also faulty. Call a subset $\mathscr{T}'$ of interactions in $\mathscr{I}_t$ *independent* if there do not exist $T, T' \in \mathscr{T}'$ with $T \subseteq T'$. In general, some interactions in $\mathscr{I}_t$ (or perhaps $\overline{\mathscr{I}_t}$) are believed to be faulty, but their number and identity are unknown. The faulty interactions cannot be identified precisely from the outcomes, *even if the full factorial design is employed*, without some restriction on their number. (Consider the situation in which every design point run fails.) We therefore suppose that a maximum number d of faulty interactions is specified.

DEFINITION 2.1 ([8]). *An array A is $(\overline{d},\overline{t})$-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 $\rho(A,\mathscr{T}_1) = \rho(A,\mathscr{T}_2) \Leftrightarrow \mathscr{T}_1 = \mathscr{T}_2$.*

If there is any set of d interactions of strength t that produce exactly the outcomes obtained when using a (d,t)-locating array A to conduct experiments, then there is exactly one such set of interactions. To avoid enumeration of all sets of d interactions of strength t, one can employ a stronger condition that for every interaction T of strength at most T and every set $\mathscr{T}_1 \subseteq \overline{\mathscr{I}_t}$ that does not contain T and for which $\mathscr{T}_1 \cup \{T\}$ is independent, it holds that $\rho(A,T) = \rho(A,\mathscr{T}_1) \Leftrightarrow T \in \mathscr{T}_1$. A locating array meeting this stronger condition is termed a *detecting array* in [8]. When using a detecting array, if there are at most d independent faulty interactions each of strength at most t, they are characterized precisely as the interactions that appear in no run that passes. We typically employ the term locating array to refer to both, but for reasons of computational efficiency the locating arrays that we use are, in fact, detecting arrays.

In practice, one does not know *a priori* how many interactions are faulty, or their strengths. Nevertheless, when responses are continuous, we can select a threshold on the responses so as to limit the number of design points yielding a "fail" outcome to locate those that make the most substantial contribution to the response. We exploit this fact later in §3.2.

2.1 A Small Example

An example is provided to demonstrate fault location, and show the limitations of covering arrays for this purpose. Suppose that we use the experimental design for five binary factors in Table 1. It is a *covering array* in which each of the $2^2\binom{5}{2} = 40$ two-way interactions is covered. A response for each design point run is listed in the adjacent column.

Table 1: Experimental design and response for each run.

		Factors					
		1	2	3	4	5	Response
Design Points	1	0	1	1	1	1	Fail
	2	1	0	1	0	0	Pass
	3	0	1	0	0	0	Fail
	4	1	0	0	1	1	Pass
	5	0	0	0	0	1	Pass
	6	1	1	0	1	0	Pass

First, let us locate faults due to main effects (*i.e.*, the individual factors or one-way interactions). The second design point run passes, so all (factor, level) pairs in it are known not to be faulty. Therefore in Table 2(a), that considers only the second design point, when factor 1 is set to one, the run is not faulty. Similarly, for factors 2, 3, 4, and 5 set to zero, one, zero, and zero, respectively. This is indicated by a check-mark (✓) in the table. Repeating to check coverage of each one-way interaction for each successful run, no single (factor, level) error accounts for the faults; see Table 2(b).

Table 2: Locating faults due to main effects.

(a) Run 2

Factors	0	1
1		✓
2	✓	
3		✓
4	✓	
5	✓	

(b) All Runs

Factors	0	1
1	✓	✓
2	✓	✓
3	✓	✓
4	✓	✓
5	✓	✓

Computing $\rho(T)$ for every one-way interaction, we obtain the sets in Table 3. Because no two sets are equal, the array is $(\overline{1},\overline{1})$-locating and when there is a single faulty one-way interaction it can be located. However, because $\{1,3,5\} \cup \{2,3,5\} = \{1,3,5\} \cup \{1,2\}$, when rows 1, 3, and 5 fail and 2, 4, and 6 pass, we cannot determine the two faulty interactions — the array is *not* $(\overline{2},\overline{1})$-locating.

Table 3: $\rho(T)$ for one-way interactions $T = \{(c,\nu)\}$.

$\nu \downarrow c \rightarrow$	1	2	3	4	5
0	{1,3,5}	{2,4,5}	{3,4,5,6}	{2,3,5}	{2,3,6}
1	{2,4,6}	{1,3,6}	{1,2}	{1,4,6}	{1,4,6}

Now, let us try to locate faults due to two-way interactions. Because the second design point run passes, all two-way interactions in it are known not to be faulty; Table 4(a) records the results. Repeating to check for coverage of each two-way interaction for each successful run, those interactions not found to pass in this way in

Table 4: Locating faults due to two-way interactions.

(a) Run 2

Factors	00	01	10	11
1, 2			✓	
1, 3				✓
1, 4			✓	
1, 5			✓	
2, 3		✓		
2, 4	✓			
2, 5	✓			
3, 4			✓	
3, 5			✓	
4, 5	✓			

(b) All Runs

Factors	00	01	10	11
1, 2	✓		✓	✓
1, 3	✓		✓	✓
1, 4	✓		✓	✓
1, 5		✓	✓	✓
2, 3	✓	✓	✓	
2, 4	✓	✓		✓
2, 5	✓	✓	✓	
3, 4	✓	✓	✓	
3, 5	✓	✓	✓	
4, 5	✓	✓	✓	✓

Table 4(b) form a set of *candidate faults*. In this example, there are nine interactions in the set of candidate faults. Now for the two-way interaction $\{(1,0),(2,1)\}$, $\rho(\{(1,0),(2,1)\}) = \{1,3\}$, and it is the only two-way interaction for which this holds; and, no one-way interaction T has $\rho(T) = \{1,3\}$. Hence if there is a single fault, it must be $\{(1,0),(2,1)\}$, and we have located the fault.

Our success for one response is not sufficient, however. Because $\rho(\{(1,0),(2,1)\}) = \{1\} = \rho(\{(2,1),(3,1)\})$, if only run 1 fails, there are at least two equally plausible explanations using only a single two-way interaction. Indeed A is *not* $(\overline{1},\overline{2})$-locating. Thus the ability to locate is more than simply coverage!

3. SCREENING AN ENGINEERED SYSTEM

We now apply locating arrays for screening in a complex engineered system. One example of a CES for which it has been particularly difficult to develop models is a *mobile ad hoc network* (MANET). A MANET is a collection of mobile wireless nodes that self-organize without the use of any fixed infrastructure or centralized control. We seek to use a locating array to screen for the influential factors and interactions on average *transport control protocol* (TCP) throughput in a simulation model of a MANET.

3.1 Designing the Experiment

We use the `ns-2` simulator [37], version 2.34, for our experimentation. Since our response of interest is average TCP throughput, we select the *file transfer protocol* (FTP) as our application because it uses TCP for reliability. We select the *internet protocol* (IP), the *Ad hoc On-demand Distance Vector* routing protocol (AODV) [42], and IEEE 802.11b *direct sequence spread spectrum* (DSSS) as protocols at the network, data link, and physical layers of the protocol stack. We also use the mobility, energy, error, and propagation models in `ns-2`. From these protocols and models we identify 75 *controllable* factors. The region of interest for each factor, *i.e.*, the range over which the factor is varied, ranges from two to ten levels, with some set according to recommendations in [33]. See Appendix A for a pointer to details of the factors and their levels.

The full-factorial design for this factor space is infeasible; it has over 10^{43} design points! In contrast, the locating array *constructed and checked manually* has only 421 design points. Except for small locating arrays [51], no general construction methods have been published. We adopted a heuristic approach to construct the LA.

Initially we selected a covering array with 75 factors and 10 levels per factor, constructed using a standard product construction [7]. We applied a post-optimization method [34] to reduce the number of levels for each factor to the desired number, eliminating rows in the process and forming an array C with 143 design points. The resulting array provides coverage of two-way interactions but does not support location. When T and T' are interactions, to distinguish them we require that $\rho(T) \neq \rho(T')$, but we ask for more, namely that $|\rho(T) \setminus \rho(T')| \geq 2$ and $|\rho(T') \setminus \rho(T)| \geq 2$; this ensures that for every two interactions of interest, there are at least two design points containing one but not the other. To accomplish this, we formed three copies of C, randomly permuted their symbols within each column, and formed their union (so that every two-way interaction is covered at least three times). The resulting array B with 429 rows turned out to be $(\overline{1},\overline{2})$-detecting. Three rows were selected by a greedy method to ensure the stronger condition that $|\rho(T) \setminus \rho(T')| \geq 2$ for every pair T, T' of interactions; then eleven rows were deleted by a greedy algorithm to remove redundant rows, ultimately producing a design with 421 rows. Appendix A gives a pointer to the locating array used as the experimental design. Our objective was not to find the smallest possible array, because a fair evaluation of the efficacy of locating arrays should not rely on substantial additional structure being present.

Ten replicates of each design point in the LA are run in `ns-2`; for each a response of TCP throughput is measured. These are averaged for each design point resulting in a vector with 421 entries of observed average TCP throughput $obsTh$.

3.2 Screening Algorithm

We describe an algorithm for screening at a high level to facilitate understanding. In each iteration of the algorithm the most significant main effect or two-way interaction is identified. These terms are accumulated in a *screening model* of average TCP throughput. However, this screening model is *not* intended as a predictive model; the quality of its current estimate allows the algorithm to select the next most significant term. The screening model is used only to identify influential main effects and two-way interactions. With its output, a predictive model can be built; see §4.

Initially, the screening model has no terms. With no other information, it should estimate the average TCP throughput to be the average of the vector of observed average throughput. This is unlikely to be a very good estimation!

Our strategy to identify the most significant factor or interaction as the term to add to the screening model is as follows. Suppose that factor F_j, $1 \leq k \leq 75$, has ℓ_j levels $L_j = \{v_{j,1}, \ldots v_{j,\ell_j}\}$. For each level ℓ, $1 \leq \ell \leq \ell_j$, of factor F_j iterate through each of the 421 design points of the locating array A. For each design point i, $1 \leq i \leq 421$, partition the contribution of the (factor F_j, level $v_{j,\ell}$) combination into one of two sets: S or $\overline{S}$. If the design point has the factor F_j set to level ℓ, *i.e.*, $a_{i,j} = v_{j,\ell}$, then add the throughput measured for design point i, $obsTh[i]$, to S; otherwise add $obsTh[i]$ to $\overline{S}$. Then, compute the (absolute) difference of the average of sets S and $\overline{S}$. (Of course, metrics other than the difference of averages could be used.) Either the difference is zero (*i.e.*, the average TCP throughput collected in the sets S and $\overline{S}$ is the same), or it is non-zero. If the difference is non-zero, then one possible explanation is that the (factor F_j, level $v_{j,\ell}$) combination is responsible for the difference.

Our hypothesis is that the (factor F_j, level $v_{j,\ell}$) combination over all combinations for which the difference between the sets is the greatest is the most significant one. If this is correct, then a term of the form $c \cdot (F_j, v_{j,\ell})$ is added to the screening model. The coefficient c is equal to the difference in average TCP throughput of each set. When this term is added to the screening model, it makes the same estimation for average TCP throughput for sets S and $\overline{S}$.

In the first iteration of this algorithm, the estimate (*i.e.*, the average of the vector of observed average TCP throughput) is used to determine deviations from each entry in the vector $obsTh$. We now have a screening model that apparently includes the most significant factor. It is now used to produce a new estimate of average TCP throughput and update the vector of residual throughput. The algorithm can be applied repeatedly to the residuals to identify the next most important factor or interaction.

While this algorithm is described for (factor, level) combinations, we actually iterate over all one-way (*i.e.*, all (factor, level) combinations) *and* all two-way interactions (*i.e.*, all pairs of (factor, level) combinations) to identify the main effect or two-way interaction of highest significance. Any number of stopping conditions may be used to decide when to terminate the model development. We use the R^2, the coefficient of determination, indicating how well data fits a line or curve; when it shows marginal improvement, we stop.

The locating array constructed for our CES is a $(\overline{d} = 1, \overline{t} = 2)$-locating array, meaning it only guarantees to be able to locate (identify) at most one $(\overline{d} = 1)$ main effect or two-way (*i.e.*, up to $\overline{t} = 2$-way) interaction. It is interesting that the LA may be used iteratively to identify subsequent significant main effects or interactions. In this sense, the algorithm uses a "heavy-hitters" approach as in compressive sensing [6].

3.3 Example of the Screening Algorithm

A small example is provided to step through one iteration of the screening algorithm. Suppose that we use the experimental design for four binary factors in Table 5. It is a covering array of strength three and therefore also a $(2, 1)$-detecting array. Factor 1 corresponds to the distribution function used for introducing errors (uniformly or exponentially distributed), factor 2 to the error rate (10^{-7} or 10^{-5}), factor 3 to the number of flows at the application layer (1 or 18), and factor 4 to the TCP packet size (64 or 2048); the levels are taken as "binary" for this example. All remaining factors are set to their default levels for experimentation. A response of observed TCP throughput for each design point, averaged over ten replicates, is listed in the column $obsTh$. (All measures are truncated to integers for simplicity.)

Table 5: Experimental design and average TCP throughput.

	Factors					
Design Points	1	2	3	4	$obsTh$	$resTh$
1	0	0	0	0	63339	-14699
2	0	0	1	1	29860	-48178
3	0	1	0	1	80801	2764
4	0	1	1	0	3804	-74234
5	1	0	0	1	373866	295828
6	1	0	1	0	3879	-74159
7	1	1	0	0	56656	-21382
8	1	1	1	1	12095	-65943

The overall mean of the $obsTh$ is 78038. Therefore, the screening model initially estimates this value for average TCP throughput, *i.e.*, $T = 78038$. The residuals ($resTh$) are computed in Table 5 by taking the difference of the observed average throughput for each design point with this initial fitted value.

Now, we iterate over each (factor,level) combination. Factor 1 is set to its low level in design points 1–4. Therefore $S = \frac{1}{4}\sum_1^4 resTh[i] = \frac{-134347}{4} = -33586$ and $\overline{S} = \frac{1}{4}\sum_5^8 resTh[i] = \frac{134344}{4} = 33586$. The absolute difference, $|S - \overline{S}| = |{-33586} - 33586| = 67172$.

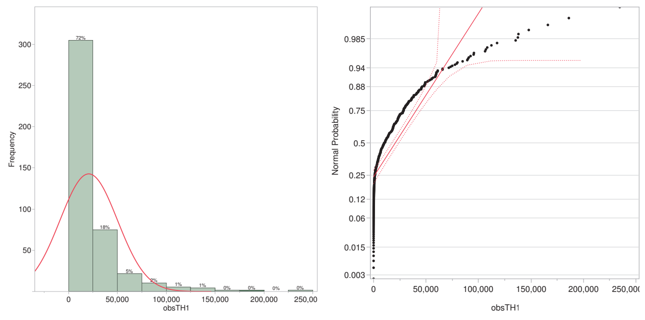
Repeating for each (factor, level) combination, as well as all two-way interactions, we find that it is a main effect that has highest absolute difference with a value of 131255. It occurs when factor 3 is set to its lowest level, namely when the number of flows at the application layer is only one. Hence we attribute this as the explanation for the largest difference and add the term $c \cdot (F_3, v_{3,0})$ to the model. The method of *ordinary least squares* (OLS) is used to fit the intercept and coefficient c of the new term. This results in an updated model of $T = 12410 + 131255 \cdot (F_3, v_{3,0})$. Its coefficient of determination is $R^2 = 0.33$.

Using this updated model, the residuals can be recomputed as input to the next iteration of the algorithm.

Next, we describe some of the obstacles arising in the practical application of the screening algorithm.

3.4 Applying the Screening Algorithm

In applying the screening algorithm to our CES, several obstacles arose. The first is that the measured average TCP throughput is not normally distributed, as Figure 2 shows; this is not uncommon in systems experimentation [12]. The best transformation of the data is a natural logarithm (Figure 3a). From the normal probability plot (Figure 3b), we find that the transformed data are still not normally distributed; nevertheless, we work with this transformation of the data.

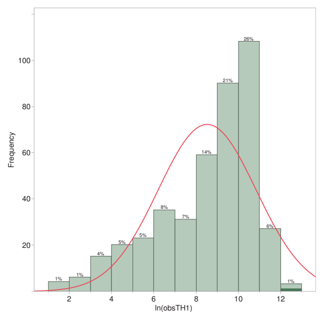


(a) Throughput distribution. (b) Normal probability plot.

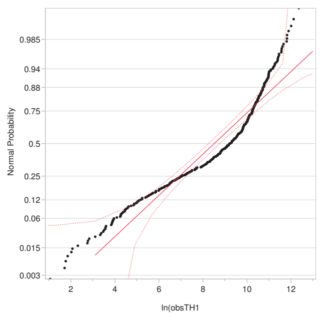
Figure 2: Distribution of the original observed average throughput, and corresponding normal probability plot.

A much larger problem arises from the fact that the LA does not cover each main effect and two-way interaction the same number of times. Indeed, binary factors are covered much more frequently (some as many as two hundred times in the 421 row LA) compared to two-way interactions of factors with ten levels (only a handful of times). This is unavoidable when one-way and two-way interactions are compared, and when factors have a different numbers of levels.

Consider the behaviour of the screening algorithm. For a binary factor the sets S and $\overline{S}$ have the same or nearly the same size and, as a result, the average of each set has small variance. In the example in §3.3, each (factor, level) combination is covered four times

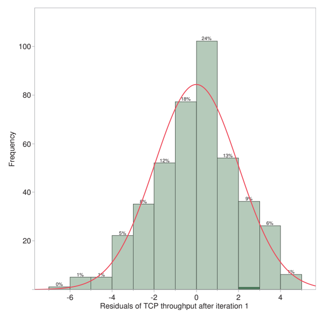


(a) ln transformation.

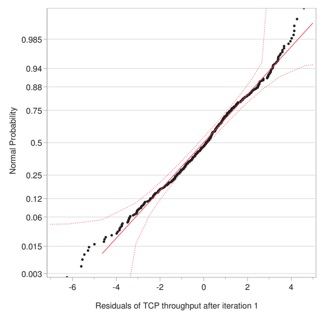


(b) Normal probability plot.

Figure 3: Natural logarithm transformation of the original observed throughput, and corresponding normal probability plot.



(a) Distribution of first residual.



(b) Normal probability plot.

Figure 4: Distribution of residuals after the first iteration of the screening algorithm, and corresponding normal probability plot.

(each column of the array has four zeros and four ones). However in general, as the number of levels for a factor increases, the size of the sets S and $\overline{S}$ may become markedly different, and the variance of the average of each set may increase greatly. Returning to the example in §3.3, the two-way interactions are not covered equally. Consider the two-way interaction $\{(1,0),(2,0)\}$. It is covered in only two rows of the array, namely $|\rho(\{(1,0),(2,0)\})| = |\{1,2\}| = 2$ (this is true for all two-way interactions in this example). Even in this small array, the coverage of two-way interactions is unbalanced resulting in S accumulating two values and $\overline{S}$ accumulating six values. This makes any direct comparison among (factor, level) combinations and/or two-way interactions impossible.

To address this problem, factors are grouped according to the number of times each level is covered in the LA; see Appendix A for a pointer to the details on how groups are formed. Now, in each iteration of the screening algorithm, the first step is to select the most significant factor or interaction from each group. Then from these candidates, the most significant factor or interaction overall is selected.

The Figure 4 shows the graphical tests for normality of the residuals after the first iteration of the screening algorithm. (Similar behaviour of the residuals is observed after each iteration.) While the figures indicate that the residuals are close to normally distributed, we check using the non-parametric Shapiro-Wilk test. This test indicates that the residuals are still not normally distributed. Hence, we use the Wilcoxon rank sum test and the Mann-Whitney U-test [14, 28, 54] to select the most significant factor or two-way interaction within each group. Then, to select the most significant factor or interaction over all groups, the Akaike information criterion (AIC_C) [1] is used.

We still need to fit the intercept and the coefficients of the terms. For a linear model with the assumptions of expected error of zero and expected variance in the error to be equal, the method of ordinary least squares (OLS) is used. However, if the expected variance in the error is unequal, OLS is no longer appropriate [32]. In this case, the method of *weighted least squares* (WLS) is used to fit the intercept and coefficients of the terms in the screening model.

3.4.1 The Resulting Screening Model

Table 6 gives the screening model for average TCP throughput developed in twelve iterations of the screening algorithm; Table 8 lists its unique factors. A Student's t-test was run on each term in the screening model and each was found to be significant; β_0 is the intercept and β_i is the coefficient of term i, $1 \le i \le 12$.

Table 6: Screening model with twelve terms.

t-Test	β_i	Factor or interaction, and level(s)
52.6	5.6	
34.5	4.4	`ErrorModel_ranvar_`$Uniform$
32.8	4.0	`ErrorModel_unit_`pkt)
-29.1	-4.7	(`ErrorModel_ranvar_`$Uniform$) * (`ErrorModel_unit_`pkt)
-11.8	-1.6	`TCP_packetSize_` 64
-12.1	-1.5	`MAC_RTSThreshold_` 0
-9.3	-1.2	`TCP_packetSize_` 128
6.5	0.9	(`TCP_RTTvar_exp_` 2) * (`TCP_min_max_RTO_` 0.1)
6.6	0.7	`TCP_min_max_RTO_` 0.2
8.4	1.1	(`ErrorModel_unit_`pkt) * (`ErrorModel_rate_` 1.0E-07)
6.3	1.1	(`ErrorModel_ranvar_`$Uniform$) * (`MAC_RTSThreshold_` 0)
5.5	0.7	`APP_flows_` 1
5.2	0.5	`RWP_Area_` 8

The first notable observation about this screening model is that it contains both main effects and two-way interactions. Moreover, it contains factors from across the layers of the protocol stack (application, transport, and MAC) and not just the transport layer; in addition, it includes factors from the error model and the mobility model. Aside from these differences with other models of TCP throughput (such as [15, 18, 30, 39–41, 55, 57, 58]), the screening model includes not just which factors or two-way interactions are significant, but the level at which each is significant.

From the statistical point of view, Table 7 shows a strong correlation among the regressors and the response of average TCP throughput. The F statistic indicates that the model is significant to the response.

Table 7: Summary statistics of the screening model in Table 6.

R^2 and Adjusted R^2: 0.84
Standard deviation: 0.92
F statistic: 180.6 on 12 and 408 df, p-value < 7.89e-155

We are encouraged by the factors and interactions identified. This includes how and into what unit errors are introduced (using a uniform distribution into packets rather than bit errors), and their interaction. Smaller sized packets (64 and 128 bytes) tend to reduce throughput. When RTS/CTS is always on (*i.e.*, the threshold is zero bytes), there is a negative impact on throughput compared to when it is configured to 1500 or 3000 bytes (always off). The retransmission timeout (RTO) and round trip time (RTT) are part of TCP's congestion control mechanism; the RTO infers packet loss by observing duplicate acknowledgements and the RTT is related to the propagation delay. The RTO is significant by itself, and in its interaction with the RTT as they work to correct and prevent network congestion. The synthetic error model of the simulator drops packets comparing them with data from an uniform distribution at a steady-state loss event rate of $1.0E$-07; this is the lowest error rate used and naturally it corresponds with higher throughput. Smaller simulation areas also result in higher throughput; a larger area has longer average shortest-hop path lengths and average higher network partition rates both of which negatively affect throughput. The throughput response is higher with fewer flows because increasing the number of flows not only may overload the network but more flows are more challenging to route in a MANET.

4. VALIDATION AND VERIFICATION

From the 75 controllable factors used in experimentation, nine unique factors are present in the twelve terms in the screening model in Table 6; these are listed in Table 8.

Table 8: Unique factors in the screening model in Table 6.

Factor	Level	
	Minimum	Maximum
TCP_RTTvar_exp_	2	4
ErrorModel_ranvar_	*Uniform*	*Exponential*
ErrorModel_unit_	*pkt*	*bit*
MAC_RTSThreshold_	0	3000
ErrorModel_rate_	$1.0E$-07	$1.0E$-05
RWP_Area_	8	40
TCP_min_max_RTO_	0.1	40
APP_flows_	1	18
TCP_packetSize_	64	2048

In order to validate the factors and interactions identified, we first conduct a full-factorial experiment for these nine factors using the extremes of their region of interest, using the statistical software JMP to analyze the results. From this, we produce a predictive model of average TCP throughput. We then examine the quality of this predictive model by comparing how it performs on random design points (*i.e.*, a design point in which the level of each factor is selected at random).

We present our validation results next.

4.1 Full-Factorial Screening in JMP

We conduct an independent 2^9 full-factorial experiment on the nine factors in Table 8. All remaining $75 - 9 = 66$ factors are fixed to their default levels. Ten replicates of each of the 2^9 design points is run, and TCP throughput measured. The results of the experimentation are input to the JMP statistical software, version 11.0 [45].

The results from the full-factorial screening experiment are given in Table 9. It includes only the main effects and two-way interactions sorted in increasing order by the p-value. The results indicate high commonality with the main effects and two-factor interactions selected by the screening algorithm that formed the screening model in Table 6. Indeed, both models have the same four most significant terms (though in a different order), and all factors and interactions in Table 6 are a subset of the terms in Table 9. Appendix A gives a pointer to the details of the predictive model for average TCP throughput that was fit using a subset of the significant terms in Table 9.

Table 9: Partial results of a 2^9 full-factorial screening experiment using JMP 11.0 on the nine factors in Table 8.

Term	**p-Value**
ErrorModel_ranvar_*ErrorModel_unit_	<.0001*
ErrorModel_ranvar_	<.0001*
ErrorModel_unit_	<.0001*
TCP_packetSize_	<.0001*
APP_flows_	<.0001*
TCP_min_max_RTO_	<.0001*
RWP_Area_	<.0001*
MAC_RTSThreshold_	<.0001*
ErrorModel_unit_*TCP_packetSize_	<.0001*
ErrorModel_rate_	<.0001*
ErrorModel_ranvar_*MAC_RTSThreshold_	<.0001*
APP_flows_*RWP_Area_	<.0001*
ErrorModel_unit_*ErrorModel_rate_	<.0001*
TCP_packetSize_*ErrorModel_rate_	<.0001*
ErrorModel_unit_*MAC_RTSThreshold_	<.0001*
ErrorModel_ranvar_*APP_flows_	<.0001*
APP_flows_*TCP_min_max_RTO_	<.0001*
ErrorModel_unit_*APP_flows_	<.0001*
ErrorModel_ranvar_*TCP_min_max_RTO_	<.0001*
ErrorModel_ranvar_*TCP_packetSize_	<.0001*
TCP_packetSize_*APP_flows_	<.0001*
TCP_min_max_RTO_*RWP_Area_	<.0001*
ErrorModel_ranvar_*RWP_Area_	<.0001*
MAC_RTSThreshold_*ErrorModel_rate_	<.0001*
TCP_min_max_RTO_*ErrorModel_rate_	<.0001*
TCP_min_max_RTO_*TCP_rttvar_exp_	0.0001
ErrorModel_unit_*TCP_min_max_RTO_	0.0001
APP_flows_*ErrorModel_rate_	0.0003
RWP_Area_*MAC_RTSThreshold_	0.0006
ErrorModel_unit_*RWP_Area_	0.001
TCP_rttvar_exp_	0.0012
TCP_packetSize_*RWP_Area_	0.002
APP_flows_*MAC_RTSThreshold_	0.0116
RWP_Area_*ErrorModel_rate_	0.0444
ErrorModel_ranvar_*TCP_rttvar_exp_	0.0515

Figure 5 shows the results of evaluating the JMP predictive model as a function of the TCP packet size, for the three levels of error rate. As in the experimentation, all remaining factors are fixed at their default levels. As expected, the results show that the highest TCP throughput is achieved when the error rate is at the lowest level ($1.0E$-07). For a given error rate the TCP throughput increases as a function of packet size, after which it decreases. An exception is for packet size 1024. Aside from this exception, these results also confirm our intuition of TCP throughput behaviour. The reason for this exception deserves further study but may be related to the default settings used for the other 66 factors not varied in this screening experiment.

We now examine the predictive accuracy of the JMP model for random design points.

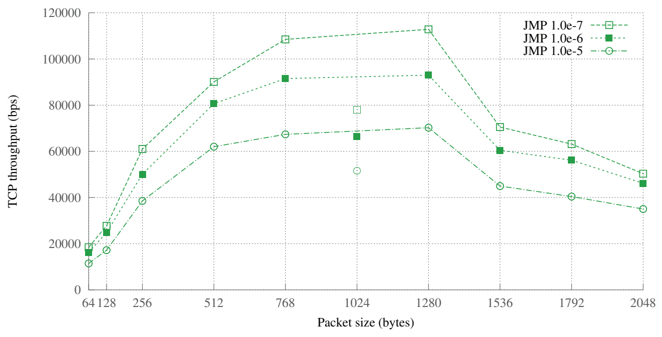


Figure 5: TCP throughput as a function of packet size as predicted the by JMP model; all other factors are at their default levels.

4.2 Predictive Accuracy of JMP Model

In order to test the predictive accuracy of the JMP model, a new experimental design of one hundred random design points is constructed. In constructing each design point, for each of factor F_j, $1 \leq j \leq 75$, a random level from L_j is selected. New mobility scenarios are also generated. Ten replicates of each of the random design points are run in the `ns-2` simulator, and the TCP throughput measured. In addition, for each experiment in the design, the JMP model is evaluated generating a new data set of fitted TCP throughput.

Figure 6 shows the average TCP throughput from simulation, and the fitted throughput from the JMP model corresponding to this random design. The mean TCP throughput from the simulations is 20,892 bps whereas the mean from the JMP model is lower, only 13,946 bps. However, the standard deviation of the results from the JMP model is smaller than the standard deviation from the simulations. Both models exhibit a few outliers. Approximately 94% of the results predicted for TCP throughput from the JMP model are in one standard deviation of the simulation results. Considering the size of the factor space, we conclude that the predicted average TCP throughput of the JMP model is similar to the average TCP throughput measured in simulation.

4.3 Predictive Accuracy of Screening Model

While the model developed in applying the screening algorithm based on the LA (Table 6) is not intended to be used as a predictive model, we were curious about its predictive accuracy. Appendix A gives a pointer to a summary of results similar to those in this section for the screening model. To our surprise, the predictive accuracy of the screening model is reasonably good. The screening model does appear to have more variability than the model developed in JMP.

5. CONCLUSIONS

Locating arrays capture the intuition that in order to see the effect of a main effect or interaction, some design point must cover it; and in order to distinguish it, the responses for the set of design points that cover it must not be equally explained by another small set of main effects or interactions. In a complex engineered system, many main effects and interactions may be significant, but our method identifies them one at a time, iteratively improving a screening model. In this way, an experimental design must be able to repeatedly locate a single "most significant" main effect or interaction. Our results show that using locating arrays for screening appears promising. Indeed while the screening targeted the identification of significant factors and two-way interactions, the screening model developed also reflects the actual behaviour well.

Despite this, the method aims only to deal with many factors and their interactions to identify the significant ones. We advocate that further experimentation is necessary after the screening is completed, both to confirm the screening results and to build a predictive model. One must be cautious not to over-fit the experimental results and claim unwarranted confidence; confirmation is needed. This is particularly a concern if the stopping criterion chosen locates too many or too few significant interactions; while our choice of R^2 appears to have worked well, future effort should address the impact of different stopping criteria. A second concern is the selection criterion for the next factor or interaction to include. Subsequent selections depend upon selections already made, so our method could in principle be misdirected by a bad selection. Our criterion of using the differences between responses for S and those for $\overline{S}$ has also worked well, but we cannot be certain that such a simple selection suffices in general. Finally, we have employed only a few locating arrays; while they have worked well in our analyses, constructing a suitable locating array remains a challenging problem that merits further research.

Certainly further experimentation is needed to assess the merit of screening using LAs, in particular on physical not just simulated complex engineered systems, and draw firm conclusions. What we can conclude is that in a challenging CES arising from a MANET, screening using locating arrays is viable and yields useful models.

Acknowledgment

Thanks to Doug Montgomery for his advice on all things statistical. This material is based in part upon work supported by the National Science Foundation under Grant No. 1421058.

6. REFERENCES

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] R. C. Bryce and C. J. Colbourn. A density-based greedy algorithm for higher strength covering arrays. *Software Testing, Verification, and Reliability*, 19:37–53, 2009.

[3] J. N. Cawse. Experimental design for combinatorial and high throughput materials development. *GE Global Research Technical Report*, 29(9):769–781, 2002.

[4] C. J. Colbourn. Combinatorial aspects of covering arrays. *Le Matematiche (Catania)*, 58:121–167, 2004.

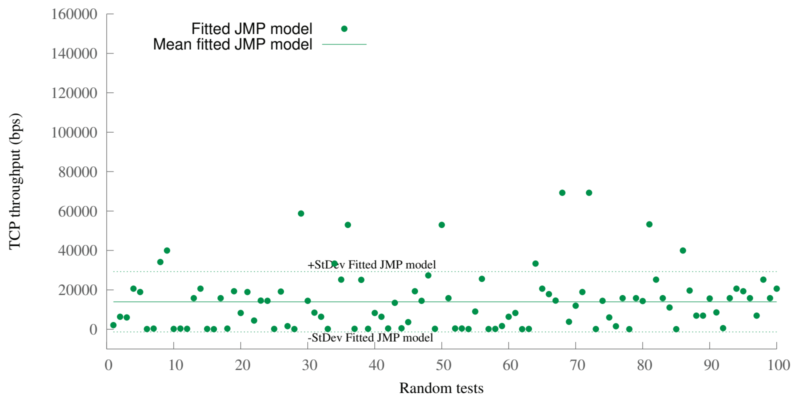
[5] C. J. Colbourn. Covering arrays and hash families. In *Information Security and Related Combinatorics*, NATO Peace and Information Security, pages 99–136. IOS Press, 2011.

[6] C. J. Colbourn, D. Horsley, and V. R. Syrotiuk. Frameproof codes and compressive sensing. In *Proc. 48th Annual Allerton Conference on Communication, Control, and Computing*, 2010.

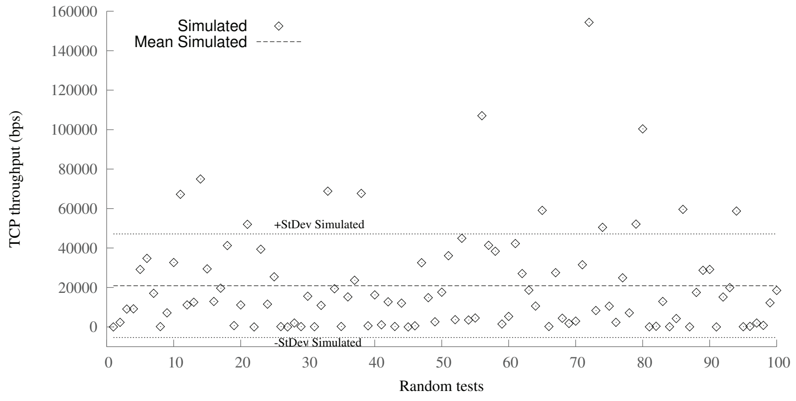
[7] C. J. Colbourn, S. S. Martirosyan, G. L. Mullen, D. E. Shasha, G. B. Sherwood, and J. L. Yucas. Products of mixed covering arrays of strength two. *Journal of Combinatorial Designs*, 14(2):124–138, 2006.

[8] C. J. Colbourn and D. W. McClary. Locating and detecting arrays for interaction faults. *Journal of Combinatorial Optimization*, 15:17–48, 2008.

[9] C. Croarkin, P. Tobias, J. J. Filliben, B. Hembree,



(a) Predictions by JMP.



(b) Simulation results.

Figure 6: Predictions by the JMP model and simulation results for random design points.

W. Guthrie, L. Trutna, and J. Prins, editors. *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST/SEMATECH, 2012.

[10] S. R. Dalal, A. J. N. Karunanithi, J. M. L. Leaton, G. C. P. Patton, and B. M. Horowitz. Model-based testing in practice. In *Proc. Intl. Conf. on Software Engineering (ICSE '99)*, pages 285–294, 1999.

[11] P. Damaschke. Adaptive versus nonadaptive attribute-efficient learning. *Machine Learning*, 41:197–215, 2000.

[12] A. B. de Oliveira, S. Fischmeister, A. Diwan, M. Hauswirth, and P. F. Sweeney. Why you should care about quantile regression. In *Proc. of the ACM Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March 2013.

[13] S. Dunietz, W. K. Ehrlich, B. D. Szablak, C. L. Mallows, and A. Iannino. Applying design of experiments to software testing. In *Proc. Intl. Conf. on Software Engineering (ICSE '97)*, pages 205–215, Los Alamitos, CA, 1997. IEEE.

[14] M. Fay and M. Proschan. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys*, 4:1–39, 2010.

[15] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications: The extended version. *SIGCOMM Computing Communications Review*, 30:43–56, 2000.

[16] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn. Refining the in-parameter-order strategy for constructing covering arrays. *J. Res. Nat. Inst. Stand. Tech.*, 113:287–297, 2008.

[17] S. Ghosh and C. Burns. Comparison of four new general classes of search designs. *Austral. New Zealand J. Stat.*, 44:357–366, 2002.

[18] K.-J. Grinnemo and A. Brunstrom. A simulation based performance analysis of a TCP extension for best-effort multimedia applications. In *Proceedings of the 35th Annual Simulation Symposium*, 2002.

[19] A. Hartman. Software and hardware testing using combinatorial covering suites. In M. C. Golumbic and I. B.-A. Hartman, editors, *Interdisciplinary Applications of Graph Theory, Combinatorics, and Algorithms*, pages 237–266. Springer, Norwell, MA, 2005.

[20] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays*. Springer-Verlag, New York, 1999.

[21] D. S. Hoskins, C. J. Colbourn, and M. Kulahci. Truncated D-optimal designs for screening experiments. *American Journal of Mathematical and Management Sciences*, 28:359–383, 2008.

[22] D. S. Hoskins, C. J. Colbourn, and D. C. Montgomery. D-optimal designs with interaction coverage. *Journal of Statistical Theory and Practice*, 3:817–830, 2009.

[23] J. P. C. Kleijnen. An overview of the design and analysis of simulation experiments for sensitivity analysis. *European Journal of Operational Research*, 164:287–300, 2005.

[24] J. P. C. Kleijnen, B. Bettonvil, and F. Persson. Screening for

the important factors in large discrete-even simulation models: Sequential bifurcation and its applications. In A. M. Dean and S. M. Lewis, editors, *Screening: Methods for Experimentation in Industry, Drug Discovery and Genetics*, chapter 13, pages 287–307. Springer-Verlag, 2006.

[25] D. Kuhn and M. Reilly. An investigation of the applicability of design of experiments to software testing. In *Proc. 27th Annual NASA Goddard/IEEE Software Engineering Workshop*, pages 91–95, Los Alamitos, CA, 2002. IEEE.

[26] D. R. Kuhn, D. R. Wallace, and A. M. Gallo. Software fault interactions and implications for software testing. *IEEE Trans. Software Engineering*, 30(6):418–421, 2004.

[27] R. Li and D. K. J. Lin. Analysis methods for supersaturated designs: Some comparisons. *Journal of Data Science*, pages 249–260, 2003.

[28] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.

[29] C. Martínez, L. Moura, D. Panario, and B. Stevens. Locating errors using ELAs, covering arrays, and adaptive testing algorithms. *SIAM J. Discrete Math.*, 23:1776–1799, 2009/10.

[30] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev.*, 27:67–82, 1997.

[31] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, Inc., 8 edition, 2012.

[32] D. C. Montgomery, E. A. Peck, and C. G. Vining. *Introduction to Linear Regression Analysis*. John Wiley & Sons, Inc., 4th edition, 2006.

[33] A. Munjal, T. Camp, and W. Navidi. Constructing rigorous MANET simulation scenarios with realistic mobility. In *European Wireless Conference (EW)*, pages 817–824, 2010.

[34] P. Nayeri, C. J. Colbourn, and G. Konjevod. Randomized postoptimization of covering arrays. *European Journal of Combinatorics*, 34:91–103, 2013.

[35] Networking and information technology research and development (NITRD) large scale networking (LSN) workshop report on complex engineered networks, 2012.

[36] C. Nie and H. Leung. A survey of combinatorial testing. *ACM Computing Surveys*, 43, 2011.

[37] The Network Simulator - `ns-2`. `http://www.isi.edu/nsnam/ns`.

[38] K. Nurmela. Upper bounds for covering arrays by tabu search. *Discrete Applied Mathematics*, 138(9):143–152, 2004.

[39] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. *SIGCOMM Computing Communications Review*, 28:303–314, 1998.

[40] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8:133–145, 2000.

[41] N. Parvez, A. Mahanti, and C. Williamson. An analytic throughput model for TCP NewReno. *IEEE/ACM Transactions on Networking*, 18:448–461, 2010.

[42] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[43] S. Poljak, A. Pultr, and V. Rödl. On qualitatively independent partitions and related problems. *Discrete Applied Math.*, 6:193–205, 1983.

[44] A. H. Ronneseth and C. J. Colbourn. Merging covering arrays and compressing multiple sequence alignments. *Discrete Applied Mathematics*, 157:2177–2190, 2009.

[45] JMP statistical software from SAS. `http://www.jmp.com`.

[46] G. Seroussi and N. H. Bshouty. Vector sets for exhaustive testing of logic circuits. *IEEE Transactions on Information Theory*, 34:513–522, 1988.

[47] D. E. Shasha, A. Y. Kouranov, L. V. Lejay, M. F. Chou, and G. M. Coruzzi. Using combinatorial design to study regulation by multiple input signals: A tool for parsimony in the post-genomics era. *Plant Physiology*, 127:1590–1594, 2001.

[48] T. Shirakura, T. Takahashi, and J. N. Srivastava. Searching probabilities for nonzero effects in search designs for the noisy case. *Ann. Statist.*, 24:2560–2568, 1996.

[49] J. N. Srivastava. Designs for searching non-negligible effects. In J. N. Srivastava, editor, *A Survey of Statistical Design and Linear Models*, pages 507–519. North–Holland, 1975.

[50] D. T. Tang and C. L. Chen. Iterative exhaustive pattern generation for logic testing. *IBM Journal Research and Development*, 28:212–219, 1984.

[51] Y. Tang, C. J. Colbourn, and J. Yin. Optimality and constructions of locating arrays. *J. Stat. Theory Pract.*, 6(1):20–29, 2012.

[52] J. Torres-Jimenez and E. Rodriguez-Tello. New upper bounds for binary covering arrays using simulated annealing. *Information Sciences*, 185:137–152, 2012.

[53] K. K. Vadde and V. R. Syrotiuk. Factor interaction on service delivery in mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 22:1335–1346, 2004.

[54] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.

[55] I. Yeom and A. L. N. Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Transactions on Networking*, 9:31–46, 1999.

[56] C. Yilmaz, M. B. Cohen, and A. Porter. Covering arrays for efficient fault characterization in complex configuration spaces. *IEEE Transactions on Software Engineering*, 31:20–34, 2006.

[57] B. Zhou, C. P. Fu, D.-M. Chiu, C. T. Lau, and L. H. Ngoh. A simple throughput model for TCP Reno. In *Proceedings of the IEEE International Communications Conference (ICC'06)*, 2006.

[58] M. Zorzi, A. Chockalingam, and R. R. Rao. Throughput analysis of TCP on channels with memory. *IEEE Journal on Selected Areas in Communications*, 18:1289–1300, 2000.

APPENDIX

A. GITHUB REPOSITORY

A `GitHub` repository provides supplementary material at: `https://github.com/locatingarray/screening.git` Specifically, it includes the 75 controllable factors in the `ns-2` simulator used in experimentation and their levels (§3.1), the 421×75 LA used as the experimental design (§3.1), a description of how factors are grouped (§3.4), the JMP model for TCP throughput, along with some statistical analysis (§4.1), and some analysis of the predictive capability of the screening model (§4.3).