A $2^k \cdot r$ factorial analysis tool for ns2measure

Claudio Cicconetti, Enzo Mingozzi, Carlo Vallati
Dipartimento di Ingegneria dell'Informazione
University of Pisa, Via Diotisalvi, 2 – 56122 Pisa, ITALY
{c.cicconetti,e.mingozzi,c.vallati}@iet.unipi.it

ABSTRACT

In recent years simulation has been the most used tool for performance analysis of communication networks, due to the ever increasing complexity of modern architectures and technologies. There are two correlated problems that are typically encountered when using simulation, especially by less experienced practitioners. Often, the number of factors that affect the response of such systems is high, which makes it difficult to distinguish their relative impact on the performance. Secondly, it would be helpful to know which are the most important factors before running the actual simulation campaign, so as to reduce the number of scenarios, hence the overall computational time, which otherwise can grow unnecessarily large. Both these problems are alleviated substantially if well-established methodologies, e.g. the $2^k \cdot r$ factorial analysis, are employed. In this work we present a new tool that implement all the procedures for running a $2^k \cdot r$ factorial analysis in a simple and sound manner. The tool, called `factorial2kr`, has been developed within the `ns2measure` framework, which is an integrated module, in a mature state of development, for collecting and analyzing statistical data with the widely used network simulator ns-2. The software is released under GPL.

D.2.6 [**Software Engineering**]; Programming environments – *performance measures;*I.6.7 [**Simulation and Modeling**]: Simulation support system – *environments*; G.3 [**Mathematics of Computing**]: Probability and Statistics – s*tatistical software.*

General Terms

Measurement, Performance, Experimentation, Verification.

Keywords

Simulation tools, ns-2, statistical analysis, network simulation, factorial analysis

1. INTRODUCTION

In the computer networking domain, the study of a system can rarely be carried out through analytical models alone. In fact, the large size and the complexity of advanced networking architectures usually lead to a considerable number of assumptions or simplifications being introduced into the mathematical model, so as to make it tractable. This irreparably increases the gap between the model and the real system, which has to be somehow recovered before the system under study goes into production. One way to achieve this goal is to build a prototype of the system, which however can be very expensive, when feasible at all, and still provides only partial results due to small-scale limitations. For these reasons, it is most often the case that simulation is used as an intermediate step between mathematical models and prototyping, because it allows to investigate the performance with an accurate level of detail under a wide variety of conditions. We now introduce the terminology that we use throughout this work in the context of simulation experiments. First, the *response variable* is the outcome of a simulation, i.e. the measured performance of the system. Second, we call *factors* the variables that affect the response variable and whose effects need to be quantified. The possible values that a factor can take are called *levels* or *values*. Then, a *simulation scenario* is a given set of factors' levels. Each scenario can be uniquely identified by all the values of its factors. Finally, a group of simulation scenarios where the primary factors are varied among a set of levels of interest identifies a *simulation campaign*.

Usually, to study a system through simulation, the analyst designs a simulation campaign where the system is emulated in different scenarios characterized by different values of the factors. In general the goal of a properly designed simulation campaign is to obtain the maximum knowledge with the minimum number of simulations [1]. Thoughtful design of the campaign also helps in separating out the effects of factors affecting the performance, and in determining if a factor has a significant or negligible effect on the response variable. The latter is not always totally obvious because the response can change with different values of a factor, but observed differences might be simply due to random variations caused by measurement errors and other parameters that were not under control.

If there are no computational constraints on the simulation campaign to be run, one can straightforwardly run one simulation for every possible combination of all the factors. This kind of analysis is called *full factorial*, and it can be automated to some extent with publicly available tools like [3], [5] and [7], which are developed for widely used network simulators, i.e. SWAN, ns-2, and openWNS. Even though automation can lessen the burden of configuring and running the experiments, still the running time of a full factorial analysis can be quite high. In fact, if there are k factors, given that the n-th factor has n_i levels, a full factorial analysis requires n experiments:

$$n = \prod_{i=1}^{k} n_i \qquad (1)$$

It is easy to see that n can grow very large even with a relatively small number of factors, due to the product in (1). Therefore, it is not always possible to run a full factorial analysis, because of constraints on the overall running time of simulations, or on the amount of output data to be collected and analyzed. In most cases, a *fractional factorial analysis* is carried out instead, where the system response to only a subset of all the possible combinations

VALUETOOLS 2009, October 20-22, 2009 - Pisa, Italy.

of factors is actually investigated. To reduce the number of required experiments, one can decrease the number of levels of each factor or remove some factors. Very often, the latter is not possible, because the analyst does not have enough insight of the system to determine *a priori* if there are factors that have a minor impact on the performance. Therefore, it is necessary to design a simulation campaign with a reduced numbers of levels in order to preliminary evaluate how each variation of the factors influences the responses. The $2^k \cdot r$ *factorial analysis* is an established method of system analysis [2], where only two levels for each factor are retained. Others can be re-introduced in subsequent phases of the analysis, based on the results obtained with this preliminary analysis. This method is well-known and widely used in practice. For instance, in [10] the authors use $2^k \cdot r$ factorial analysis to evaluate the performance of a novel routing protocol for mobile ad-hoc wireless networks. In [11] this method is used to identify the parameters that influence the performance of the video transmission over frame relay networks. The $2^k \cdot r$ factorial analysis is not only useful in networking but there are several works that exploit it in different fields, e.g. in [12] it is used to investigate the impact of the design parameters of a chip packaging process.

In this paper we present a tool, called `factorial2kr`[1], that allows to perform a $2^k \cdot r$ factorial analysis starting from the data of a simulation campaign appropriately designed. Our tool helps to automate the analysis process and to speed up the analysis phase. Moreover the automation of post-processing simulation output analysis helps to increase the credibility of the results reducing human errors, as discussed in [4]. The tool is developed inside the framework `ns2measure` [8], which is a statistical environment that can be used to collect data from ns2 simulations. The standard method for carrying out a $2^k \cdot r$ factorial analysis would be to either write a home-made tool or import the simulation data, after post-processing and filtering, into a numerical computational environment having such feature, like Matlab[2]. Both approaches are error-prone, since some operations have to be carried out manually. On the other hand, with our proposed solution, it is possible to carry out the whole process of running simulations with ns-2[3] and performing a $2^k \cdot r$ factorial analysis on the output data in an *integrated* manner, hence without the possibility of introducing human errors due to, e.g., incomplete or mismatched data, erroneous conversion from one data format to another.

The rest of this document is organized as follow. Section 2 contains a detailed description on the $2^k \cdot r$ factorial analysis methodology. Section 3 illustrates briefly the `ns2measure` framework, in the context of which we developed the `factorial2kr` tool. Section 4 describes the tool, while a usage example is included in Section 5. Finally the Section 6 draws the conclusions.

2. $2^k \cdot r$ FACTORIAL ANALYSIS

The $2^k \cdot r$ factorial analysis helps in separating out the effects of all factors obtaining for each one the relative and absolute contributions in terms of a performance index. It needs a simulations set in which the k parameters to be investigated are fully varied between two fixed value (one *low* and one *high*, respectively) to evaluate the contribution of every factor. Every scenario is replicated r times, each time with different seeds for initializing the random number generators, to evaluate the contribution of the statistical variations and experimental errors. The results of this analysis are the absolute and relative contributions of all factors and of any possible combination of them. The combination of two or more factors is the effect due to the mutual variation of these factors. An absolute contribution represents the variation of the performance index in absolute terms with respect to the corresponding parameter: a positive value indicates an increment of the object function with a higher value of that factor. A relative contribution is a percentage that indicates the slice of the total shift of performance index due to the variation of the value of a parameter from the low to the high value. The variations of the results that cannot be considered to depend on the factors, but might be due to statistical errors or parameters not considered into the analysis, fall into the measure called *unexplained variations*. Starting from the value of the unexplained variations, the confidence interval of the absolute contributions is calculated; this interval helps to quantify the reliability of the computed contributions.

In the following we are going to illustrate how the contribution are evaluated starting from the responses of all replicas in each scenario.

The analytical model used to represent the response of any scenario in the simple $k = 2$ case is the following:

$$y = q_0 + q_a \cdot x_b + q_b \cdot x_b + q_{a \cdot b} \cdot x_a \cdot x_b + e$$

where q_0 accounts for the amount of response that does not change between the different scenarios (a kind of baseline), whereas the other q_i terms represent each the contribution of a different factor i, x_i is either 1 or –1 depending on whether the level is low or high, respectively, and e accounts for the experimental error. Moreover, the interferences due to interactions between factors are also taken into account: two, or more, factors are said to interact if the amount of effect of one depends upon the level of the others. The term q_{ij} accounts for the combined effect of factors i and j.

Every absolute effect is calculated from the mean responses using the following formula:

$$q_j = \frac{1}{2^k} \sum_{i=1}^{2^k} S_{i,j} \cdot \bar{y}_i$$

where $\bar{y}_i$ is the average result of the i-th scenario and S is a matrix called sign table; an example of it is the following (in the two-factor scenario):

$$S = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

This matrix is formed using the following guidelines. The first column of the matrix consists of all 1's and is used to calculate the average of all scenario results, the next two columns represent a single parameter and contain basically all possible combination of –1 and 1. The fourth column is the product of the entries in the

[1] http://cng1.iet.unipi.it/wiki/index.php/Factorial2kr

[2] http://www.mathworks.com/matlabcentral/fileexchange/9448

[3] http://www.isi.edu/nsnam/ns/

previous columns and corresponds to the combined effect of the two parameters. Each row has associated a single scenario in which the parameters and the value of each element in the row is 1 if the value of the corresponding factor is high, otherwise it is –1.

The q_i's are absolute contributions; if we want to derive the relative contributions, we have to compute the variations. Specifically, the total variation or Total Sum of Squares (SST) is given by:

$$\sum_{i,j}\left(y_{i,j}-\bar{y}\right)^2$$

where $\bar{y}$ is the mean of responses from all replications of all experiments, while y_{ij} is the result of the j-th replica of the i-th scenario. The SST can be divided into several parts as follows:

$$\sum_{i,j}\left(y_{i,j}-\bar{y}\right)^2 = 2^k \cdot r \cdot q_a^2 + ... + 2^k \cdot r \cdot q_{a \cdot b}^2 + ... + \sum_{i,j} e_{i,j}^2$$

$$\mathrm{SST} = \mathrm{SSA} + \ldots + \mathrm{SSAB} + \ldots + \mathrm{SSE}$$

where $e_{i,j} = y_{i,j} - q_0 - q_a \cdot x_a - ... - q_{ab} \cdot x_a \cdot x_b - ...$, and SSA, SSB, SSAB are variations explained by factors A, B and interaction AB, respectively. The SSE is the unexplained variation attributed to the experimental errors and parameters that were not under control.

Since the total variation can be represented by the following:

$$SST = \sum_{i,j}\left(y_{ij}-\bar{y}\right)^2 = \sum_{i,j} y_{ij}^2 - \bar{y}^2 = \sum_{i,j} y_{ij}^2 - q_0^2 = SSY - SS0$$

we can write this relation as SST = SSY – SS0 = SSA + SSB + … + SSAB + … + SSE. This can be used as a lead to compute SSE. In conclusion, the sum of squares is computed using the following equations:

$$SSY = \sum_{i=1}^{2^k}\sum_{j=1}^{r} y_{ij}^2$$

$$SS0 = 2^k \cdot r \cdot q_0^2$$

$$SST = SSY - SS0$$

$$SSj = s^k \cdot r \cdot q_j^2 \quad \text{where } j = 1, 2, ..., 2^{k-1}$$

$$SSE = SST - \sum_{j=1}^{2^k-1} SSj$$

From the sum of squares we can obtain the percentage of the variation of y due to the j-th effect evaluating the ratios between the sum of squares of the factors and the total sum of squares.

Confidence intervals for the effects can eventually be derived by computing the variance of errors, which is estimated from the SSE as follows:

$$s_e^2 = \frac{SSE}{2^k \cdot (r-1)}$$

The quantity on the right side of this equation is called Mean Square of Errors (MSE). The variance of the effects can be easily estimated from MSE:

$$s_{q_0} = s_{q_a} = s_{q_b} = ... = s_{q_{ab}} = ... = \frac{s_e}{\sqrt{2^k \cdot r}}$$

The confidence intervals for the effects are:

$$q_i = \mp t_{\left[\frac{1-\alpha}{2};2^k \cdot (r-1)\right]} \cdot s_{q_i}$$

where the t-value is read at $2^k \cdot (r-1)$ degrees of freedom. The confidence interval can be used to understand if a factor can be ignored or not. Starting from the assumption that the statistical errors must be negligible in a well designed simulation campaign, if the range of the absolute contribution of one factor contains the zero, the effect of that factor can be ignored.

The statistical confidence of the results is based on the following assumptions, which ensure that the observations are independent and normally distributed with constant variance:

(i) the model errors are statistically independent;

(ii) the errors are normally distributed;

Visual tests can be run *a posteriori* on the results to check whether the assumptions hold with an acceptable approximation:

(i) (independent errors) We compute the residuals and prepare a scattered plot of residuals versus the predicted response. Any visible trends in the scatter plot indicate a dependence of errors on the factors levels. If the residuals are one or more orders of magnitude smaller than the predicted response, any trend can be ignored.

(ii) (normally distributed errors) We prepare normal quantile-quantile (Q-Q) plots of errors. If any plot is approximately linear, the assumption is satisfied.

A full example analysis is reported in Section 5 to illustrate how to use the method in practice, also including the visual test plots to verify the validity of the assumptions.

3. `ns2measure` FRAMEWORK

This section describes the `ns2measure`[4] module [8], distributed under the GNU Public License (GPL), which contains the `factorial2kr` tool. The software `ns2measure`, released as a patch for ns-2, addresses two problems: the collection of samples of metrics and the statistical analysis of output data. As for data collection, it provides a general mechanism for specifying which events are to be logged. This allows one to record data about any type of event rather than just data related to packet transmission events (which, as already said, form ns-2 traces). Data collection is performed efficiently, avoiding frequent I/O, which would otherwise slow down the simulation. The format used in the data log entries simplifies the extraction of information for post-processing analysis, e.g. for generating plots. The data collection subsystem is based on the implementation of a C++ class called `Stat`, which processes and organizes samples from an arbitrary number of different metrics. When the user instruments the ns-2 C++ code with calls to a `Stat::put()` method, samples of a metric are passed to a `Stat` object. The samples are processed into a different histogram for each metric and only the final outcome is written to file. The `Stat` class provides support for three types of data: metrics averaged over time (e.g. throughput or loss rate),

[4] http://cng1.iet.unipi.it/wiki/index.php/Ns2measure

Table 1. Example output of analysis $2^k \cdot r$.

Base value	*y*		
Unexplained variations	*err*		
Factor / Combination	**Absolute contribution**	**Confidence interval**	**Relative contribution**
A	x_A	c_A	p_A
B	x_B	c_B	p_B
C	x_C	c_C	p_C
A*B	x_{AB}	c_{AB}	p_{AB}
A*C	x_{AC}	c_{AC}	p_{AC}
B*C	x_{BC}	c_{BC}	p_{BC}
A*B*C	x_{ABC}	c_{ABC}	p_{ABC}

metrics reflecting stochastic values over continuous-time (e.g. number of packets in a queue), and metrics reflecting stochastic values over discrete-time (e.g. end-to-end delay for a flow of packets).

As for automating statistical analysis, `ns2measure` allows a user to execute a number of independent replications of the same simulation scenario and to compute means and confidence intervals on the chosen metrics. This framework relieves the ns-2 user from having to write code i) in the Tcl scenarios for selecting independent sub-streams of random numbers (which is seldom done rigorously), and ii) in some post-processing scripting language for computing confidence intervals (which is seldom done at all). However, `ns2measure` still leaves it to the user to define simulation scenarios. When a large set of such scenarios, which differ from one another by few lines of code that modifies a single factor, are to be generated, things rapidly get out of control of the simulation user, jeopardizing the credibility of the whole campaign. To solve this problem recently we have improved the `ns2measure` framework adding a group of tools, called `ANSWER`, that helps to automate the simulation workflow explicitly designed for facilitating large-scale simulation campaigns, i.e. those involving many factors [7]. The use of this tools reduces the space for errors when defining scenarios, controls the execution of a large number of scenarios, and reduces the time overhead required for output data analysis.

4. `factorial2kr` TOOL

The factorial2k tool is released under GPL and can be downloaded at:

http://cng1.iet.unipi.it/wiki/index.php/Factorial2kr

As described below, it performs the post simulation analysis using the factorial methodology described in general terms in Section 2.

The first step is to run a simulation campaign designed according to the $2^k \cdot r$ factorial guidelines: two only levels for each of the k factors (one high and one low) are mutually varied all together obtaining all the possible combinations. Each combination represents a scenario that must be replicated with different random seeds r times. Starting from the raw data collected using `ns2measure`, the `factorial2kr` tool analyzes the results performing all the passages previously illustrated finally showing as output the absolute and relative contributions of each factor

Table 2. Values of the simulation parameters.

Parameter	**Value**
Air interface	IEEE 802.11a OFDM
Slot Time	9 μs
SIFS	16 μs
Physical header	192 μs
Channel rate	6 Mbps
RTS/CTS	Disabled
Physical model	Ideal channel
Traffic	Saturation mode

and their combinations. Moreover, the tool uses the results of the single replicas to evaluate the statistical errors, which are summarized by the confidence interval of the absolute contributions and the percentage of the unexplained variations. The obtained output is shown in Table 1.

The interpretation of the output is as follows. Using the same notation as in Section 2, the base value of the response variable is *y*. Each factor, or combination of factors, *i* contributes to the final value of the metric by an amount equal to x_i, $i \in$ {A, B, C, AB, AC, BC, ABC } in absolute terms. For instance, if $x_A > 0$, this means that the metric of interest increases by x_A by switching the level of factor A from low to high. Similarly, if $x_{AB} < 0$, this means that the metric of interest decreases as factors A and B switch from low to high level *together*. Note that this is in addition to any change of the metric of interest due to each single factor. The relative weight of each factor, or combination of factors, is reported in the last column as a percentage of variation. The rows with the highest values of p_i correspond to the most important factors, as far as the metric of interest is concerned. Note that $\sum p_i = 100 - e$, i.e. in general there is a fraction of the variation which cannot be explained by changing the values of the factors. This unexplained variation is the sum of two components: i) errors due to the initialization of the random number generators and ii) errors due to the system parameters not considered as factors in the analysis.

All the passages performed according to the guidelines of the factorial methodology are based on the assumptions of independent and normally distributed errors. Such assumptions must be verified to validate the analysis. To this purpose, the tool provides, in two different files, the raw data to plot the residuals versus the predicted response and the normal quantile-quantile (Q-Q).

5. EXAMPLE SCENARIO

In this section we perform a full $2^k \cdot r$ factorial analysis in an example scenario to better illustrate how the method can be used in practice. For this reason, we selected on purpose a system that has been studied deeply in the last years through both simulations and analytical models. This way, we can focus on showing how to interpret the data coming from the `factorial2kr` tool under known circumstances.

The system under study consists of a single-hop IEEE 802.11 wireless network with a number of stations, all in the transmission range of one another, transmitting data in asymptotic conditions, i.e. a constant size packet is generated at the Medium Access Control (MAC) layer as soon as the previous one is successfully dis-

Table 3. Factors and their values.

Primary factor	Symbol	Low	High
Number of stations	N_{sta}	10	50
Number of maximum retransmissions	R_{max}	2	7
Packet size	P_{size}	100	1000
Minimum back-off window size	T_{min}	7	63
Maximum back-off window size	T_{max}	63	1023

patched. No channel errors are introduced. Such a system has been modeled mathematically in the popular work of Bianchi [9]. In Table 2 we illustrate the system parameters that do not change during the factorial simulation campaign. Since asymptotic conditions cannot be simulated with ns-2, we have enhanced the latest version of the simulator (2.33) with this feature. A patch to the simulator is provided, together with the scripts to run the simulation campaign and the full output data, in the `factorial2kr` website. A more detailed usage of the tool can be found in appendix, while the xml configuration file of the following simulation campaign in the ANSWER format [8] can be found in Figure 4.

We have focused our attention on the contention mechanism that each station uses to gain a transmission opportunity. Table 3 shows the factors with the relative low and high values used for the $2^k \cdot r$ factorial analysis: minimum and maximum size of the back-off window size; maximum number of retransmissions before a packet is dropped; packet size; number of stations. Note that the latter is not a factor under control of the system designer. However, its presence into the factorial analysis helps to evaluate how the system behavior change with different levels of connected stations and how each factor changes its influence with an increasing network load.

The metrics, or response variables, considered are defined as follows. The *throughput* is the amount of data that each station delivers successfully to destination, without including the MAC and physical headers, in the unit of time. The *average end to end delay* is the average time that elapses between when a packet starts the back-off algorithm to gain access to the medium and when the station receives the acknowledgement after its successful transmission. The *packet drop rate,* is the number of packets dropped in the unit of time. Since we have assumed ideal channel conditions, a packet can be dropped only if the maximum number of retransmissions is exceeded. Finally, the *average number of retransmissions* is the average number of times that a packet has to be retransmitted until either an acknowledgment is received by the sending station or it is dropped.

The `factorial2kr` tool has been run once for each metric. A broad view of the overall results is illustrated in Figure 1, which contains the four histograms of the relative contributions, one for each response variable. Only the factors, and their combinations, that affect the performance most are reported. Once these major contributors are selected, an analyst can understand how they influence the response variable by looking at the sign of the absolute contributions. If an absolute contribution of a factor is positive, it means that its transition from the low value to the high increases the response variable; on the contrary, if the response is negative, the transition from low to high decreases the response.

In the rest of the analysis we focus on a single metric, i.e. the throughput, whose complete `factorial2kr` output is reported in Table 4. As expected, the throughput is influenced mostly by the number of terminals, the packet size and their combination. Regarding the number of stations, when it increases from 10 to 50 the amount of transmitted data decreases by 60% of the overall variation because the bandwidth is shared by an increasing number of competitors, and the probability of collision also increases. On the other hand, the throughput increases with the packet size because the impact of the MAC overhead becomes relatively smaller.

The percentage of the errors is approximately zero for all the response variables and therefore the confidence intervals for the absolute values are negligible. As last step, we have performed the visual tests in order to verify that the assumptions of independent and normally distributed errors hold. Figure 2 shows the scattered plot of residual versus the predicted response, used to verify that the errors are normally distributed. As can be seen this assumption holds because it does not show any visible trend. Figure 3 shows the normal quantile-quantile plot of errors, used to verify that the errors are normally distributed. As it is shown, the plot is approximately linear and therefore the initial assumption is true with a good approximation.

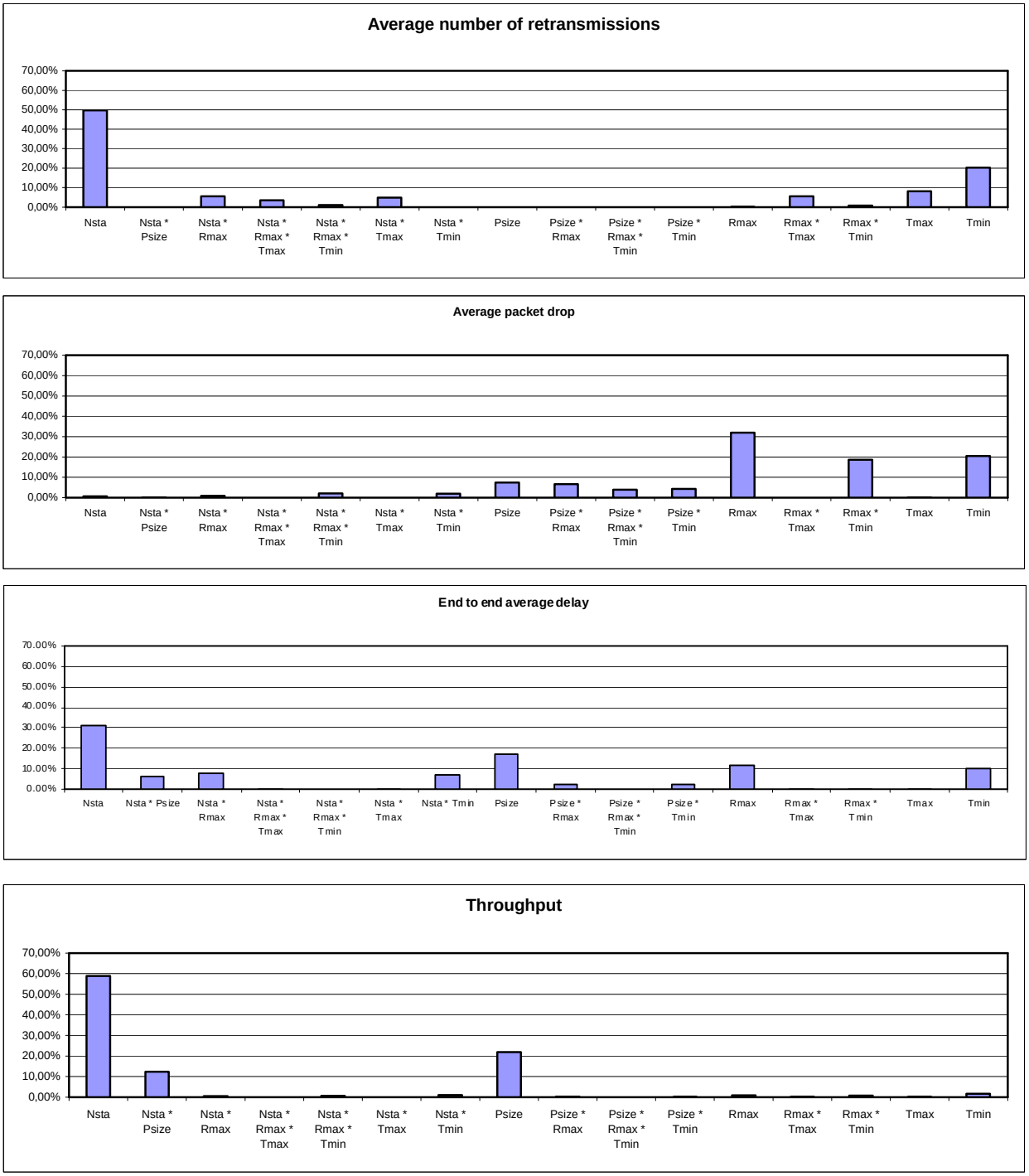

Figure 1. Factorial analysis relative contributions.

Table 4. Throughput factorial analysis results.

Metric base	65437.47 (bytes/sec)	
Confidence interval	±59.257222	
Factors (or their combinations)	**Absolute contributions**	**Relative contributions**
Nsta	-48975.2	58.97%
Psize	29840.1	21.89%
Nsta*Psize	-22432.7	12.37%
Tmin	8226.747	1.66%
Nsta*Tmin	-6584.51	1.07%
Rmax	5849.969	0.84%
Rmax*Tmin	-5274.95	0.68%
Nsta*Rmax*Tmin	4831.212	0.57%
Nsta*Rmax	-4428.76	0.48%
Psize*Tmin	4422.537	0.48%
Nsta*Psize*Tmin	-3674.17	0.33%
Psize*Rmax	2565.388	0.16%
Psize*Rmax*Tmin	-2211.61	0.12%
Nsta*Psize*Rmax*Tmin	2008.121	0.10%
Nsta*Psize*Rmax	-1930.88	0.09%
Tmax	1678.856	0.07%
Rmax*Tmax	1212.083	0.04%
Psize*Tmax	839.4812	0.02%
Nsta*Tmin*Tmax	774.3041	0.01%
Rmax*Tmin*Tmax	-637.067	0.01%
Psize*Rmax*Tmax	567.0042	0.01%
Nsta*Rmax*Tmin*Tmax	562.2997	0.01%
Nsta*Psize*Tmin*Tmax	215.6077	0.00%
Psize*Rmax*Tmin*Tmax	-213.229	0.00%
Nsta*Psize*Rmax*Tmin*Tmax	185.5881	0.00%
Tmin*Tmax	-170.294	0.00%
Nsta*Rmax*Tmax	-159.852	0.00%
Nsta*Psize*Rmax*Tmax	-108.347	0.00%
Nsta*Psize*Tmax	-78.3278	0.00%
Psize*Tmin*Tmax	59.2475	0.00%
Nsta*Tmax	52.15197	0.00%
e		0.00%

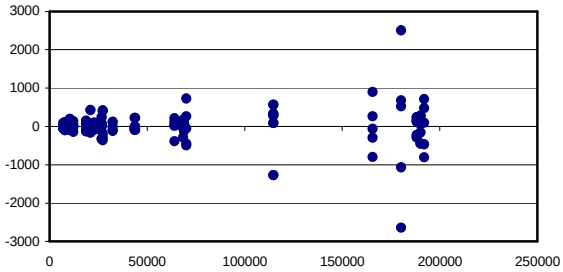

Figure 2. Plot of residuals versus the predicted response.

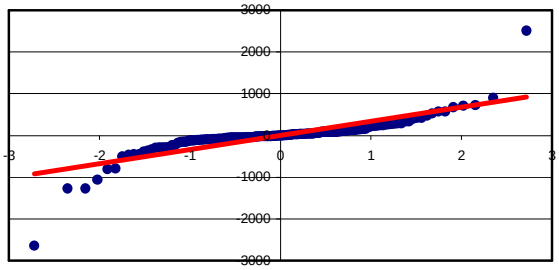

Figure 3. Plot of normal quantile-quantile.

6. CONCLUSIONS

The $2^k \cdot r$ factorial analysis is a well known method to find out the parameters that most influence the performance of a system, useful for optimally designing the experiments. In this work we have presented a tool called `factorial2kr` that in conjunction with `ANSWER` helps to completely automate the process to perform a factorial analysis. Our work automates the launch of the simulation campaign and analyzes the obtained results helping the end user to find out the contributions that modifies mostly the system responses.

REFERENCES

[1] Law, A. M. and Kelton, W. D. Simulation modeling and analysis. Third edition, McGraw-Hill, 2000, ISSN 0070592926

[2] D. C. Montgomery, “Design and Analysis of Experiments,” 6th edition, Wiley ed., ISBN: 978-0-471-48735-7, December 2004.

[3] Perrone, L. F., Kenna, C. J., and Ward, B. C. 2008. Enhancing the Credibility of Wireless Network Simulations with Experiment Automation. In Proceedings of the 2008 IEEE international Conference on Wireless & Mobile Computing, Networking & Communication, WIMOB.

[4] Perrone, L.F., Cicconetti, C. ,Stea, G. ,Ward, B., On The Automation of Computer Network Simulators. In proceedings of SIMUTOOLS 2009.

[5] Bültmann, D., Muehleisen, M., Schinnenburg, M., Klagges, K., openWNS - open Wireless Network Simulator. In proceedings of European Wireless 2009.

[6] Ewing, G., Pawlikowski, K., McNickle, D., Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation. In proceedings the 13th European Simulation Multiconference, ESM'99.

[7] Andreozzi, M. M., Stea, G., and Vallati, C. 2009. A framework for large-scale simulations and output result analysis with ns-2. In proceedings of SIMUTOOLS 2009.

[8] Cicconetti, C., Mingozzi, E., and Stea, G. 2006. An integrated framework for enabling effective data collection and statistical analysis with ns-2. In Proceeding From the 2006 Workshop on Ns-2: the IP Network Simulator, WNS2 '06.

[9] Bianchi, G., Performance analysis of the IEEE 802.11 distributed coordination function, Selected Areas in Communications, IEEE Journal on , vol.18, no.3, pp.535-547, Mar 2000.

[10] Cano, J.-C.; Manzoni, P.; Sanchez, M., Evaluating the impact of group mobility on the performance of mobile ad hoc networks. In proceedings of 2004 IEEE International Conference on Communications.

[11] Sharon, C.M.; Lambadaris, I.; Devetsikiotis, M.; Kaye, A.R., Modeling and control of VBR H.261 video transmission over frame relay networks. Circuits and Systems for Video Technology, IEEE Transactions on, Volume 7, Issue 3, Jun 1997 Page(s):527 – 538.

[12] Lee, C.-C.; Chang, S.-M.; Chiang, K.-N., Sensitivity Design of DL-WLCSP Using DOE With Factorial Analysis Technology. Advanced Packaging, IEEE Transactions on, Volume 30, Issue 1, Feb. 2007 Page(s):44 – 55.

APPENDIX: `factorial2kr` USAGE

The `factorial2kr` takes as input the raw data files where `ns2measure` saves the simulation results and one configuration file that describes the scenario. The raw data files obtained from a simulation campaign are one for each scenario and they contain the results of each response variable for all the replicas of a particular combination of dynamic factors. They contain no meta information about the simulations and therefore they must be paired with a description file in order to be processed by the factorial tool. The usage of the tool is described below:

```
 factorial2kr path_config_file
-c conf (specify the confidence level def = 0.90)
-r name_file (save data for residual visual test)
-q name_file (save data for quantile visual test)
-o name (use a different the response variable)
-d name_dir (specify the directory of savefiles)
-n id (id run to use)
```

Basically the tool takes as input only the path to the configuration file that contains all the necessary information needed to perform the analysis. Some parameters, like the directory where the output data are stored and the name of the response variable, can be specified also as command-line arguments. The configuration file is mandatory in any case and its structure is the following:

```
savefile_dir dir_savefiles
response_var name_response
num_pr_factors num_factors
fact_1
fact_2
fact_3
fact_4
file_scenario_1 fact_1 0 fact_2 0 fact_3 0 fact_4 0
file_scenario_1 fact_1 1 fact_2 0 fact_3 0 fact_4 0
..
file_scenario_1 fact_1 1 fact_2 1 fact_3 1 fact_4 0
file_scenario_1 fact_1 1 fact_2 1 fact_3 1 fact_4 1
```

The configuration file contains at the top the directory with the results, the name of the response variable, a list of the factors and a description of all the data files. Each file represents a single scenario and must be followed by the list of the factors with a 0 or a 1 respectively if in the corresponding scenario the factor is at the low or high value.

Starting from the configuration file and the simulations data the tool provides the following results of the factorial analysis, where CI is the confidence interval half-width, AC is the absolute contribution value, and RC is the relative contribution value:

```
name_response: baseline [+-CI]
fact_1: AC [+-CI],per=RC %
fact_2: AC [+-CI],per=RC %
fact_3: AC [+-CI],per=RC %
fact_4: AC [+-CI],per=RC %
fact_1*fact_2: AC [+-CI],per=RC %
fact_2*fact_3: AC [+-CI],per=RC %
…
fact_1*fact_2*fact_3*fact_4: AC [+-CI],per=RC %
errors per: errors %
```

First comes the name of the considered metric and its baseline. For each factor the absolute and the relative contributions are shown. This last is an indicator of how much the variation of that factor influences the response variable, the percentage represents the slice of the total response variation directly connected with one factor. At the end the percentage of unexplained variations is provided. Starting from this value, the confidence interval is evaluated and shown for each contribution.

If the tool is launched with the options –r and –q, it saves into the specified files the raw data to plot the visual tests graphs.

In order to automate all these stages we have published on [7] a modified version of launcher that can provide the results of a factorial analysis directly from simulation data without the direct use of the `factorial2kr`. To launch a factorial campaign, the user has to write an xml configuration file which describes the scenarios. An example can be found in Figure 4.After that, the simulations can be run using the following command:

launch.py run sim_desc.xml

Afterwards, the resulting data can be automatically analyzed using the factorial2kr tool through the following statement:

launch.py fact sim_desc.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<simulation>
        <name>wifi</name>
        <description>Wifi simulations </description>
        <check_path>path_to_check</check_path>
        <recover_path>path_to_recover</recover_path>
        <factorial2kr_path>path_to_factorial2kr</factorial2kr_path>
        <ns_path> path_to_ns </ns_path>
        <ns_output> ns_dir_output </ns_output>
        <base_scenario>wifi.tcl</base_scenario>
        <min_run>5</min_run>
        <max_run>5</max_run>
        <savefile_dir>savefile</savefile_dir>
        <factorial_response>wifi_avg_retx wifi_e2e_delay_a wifi_pkt_drop </factorial_response>
        <factorial2kr_save>fact</factorial2kr_save>
        <output_dir>output</output_dir>
        <check_metrics>wifi_avg_retx wifi_e2e_delay_a wifi_pkt_drop wifi_tpt</check_metrics>
        <check_conf_level>0.95</check_conf_level>
        <check_th>0.05</check_th>
        <multicell>
                <param>
                        <name>num-stations</name>
                        <tclname>n-sta</tclname>
                        <value alias="" value="10" level="low"></value>
                        <value alias="" value="50" level="high"></value>
                </param>

                <param>
                        <name>packet-size</name>
                        <description>Packet size</description>
                        <tclname>pkt-size</tclname>
                        <value alias="" value="100" level="low"></value>
                        <value alias="" value="1000" level="high"></value>
                </param>

                <param>
                        <name>max-retx</name>
                        <description>Number of retransmissions</description>
                        <tclname>n-retx</tclname>
                        <value alias="" value="2" level="low"></value>
                        <value alias="" value="7" level="high"></value>
                </param>

                <param>
                        <name>min-cwnd</name>
                        <description>Min contention window backoff</description>
                        <tclname>cw-min</tclname>
                        <value alias="" value="7" level="low"></value>
                        <value alias="" value="63" level="high"></value>
                </param>

                <param>
                        <name>max-cwnd</name>
                        <description>Max contention window backoff</description>
                        <tclname>cw-max</tclname>
                        <value alias="" value="63" level="low"></value>
                        <value alias="" value="1023" level="high"></value>
                </param>
        </multicell>
</simulation>
```

Figure 4. XML configuration file.