# A Parameter Optimization Tool and Its Application to Throughput Estimation Model for Wireless LAN

Nobuo Funabiki[1(✉)], Chihiro Taniguchi[1], Kyaw Soe Lwin[1], Khin Khin Zaw[1], and Wen-Chung Kao[2]

[1] Department of Electrical and Communication Engineering, Okayama University, Okayama, Japan
funabiki@okayama-u.ac.jp
[2] Department of Electrical Engineering, National Taiwan Normal University, Taipei, Taiwan
jungkao@ntnu.edu.tw

**Abstract.** Most of algorithms or logics to solve mathematical problems adopt multiple parameters such as coefficients and thresholds whose values should be carefully selected to enhance the performance. Conventionally, their developers or users tune them by trial and error, where they repeat to apply algorithms or logics with proper parameter values to available input data sets and modify them based on the results. As a result, this tuning process may consume a great deal of energy and not always provide the optimal ones. In this paper, we present a versatile *parameter optimization tool* to explore optimal values of parameters for various algorithms/logics. We show the effectiveness through applications for the *throughput estimation model* for wireless local-area networks, with the extension of the model.

## 1 Introduction

Most of algorithms or logics to solve mathematical problems have multiple parameters such as coefficients and thresholds whose values may critically affect their performance and thus should be carefully selected. Conventionally, it will take a considerable of trials and errors for developers or users to tune the values. They will select the proper values for the parameters and apply the algorithm/logic using them to available input data sets. Then, by evaluating the results, they modify the values of specific parameters and repeat the process. As a result, the tuning process of parameters may consume a great deal of energy, and may not always provide the optimal ones, because the results principally depend on adopting input data sets.

In this paper, we present a *parameter optimization tool* that can be extensively used for a variety of algorithms or logics to explore their optimal parameter values. For now, the target algorithm or logic whose parameters are optimized

This work is partially supported by JSPS KAKENHI (16K00127).


L. Barolli and O. Terzo (eds.), *Complex, Intelligent, and Software Intensive Systems*, Advances in Intelligent Systems and Computing 611, DOI 10.1007/978-3-319-61566-0_65

by the proposed tool is called the *logic*, and the program to implement a logic is the *logic program*. The proposed tool is separated from the logic program so that the developers of logic programs can implement them independently, if they follow the few conditions to use this tool, which will be discussed later.

This tool adopts a local search method to detect optimal parameter values. This method jointly uses the *tabu table* and the *hill-climbing function* to avoid a local minimum convergence [1,2]. More specifically, this algorithm repeats three phases: (1) to prepare three sets of parameters whose values are slightly changed between them by following the given specification, (2) to apply the logic program with them using input data sets, and (3) to select the best parameter value set among them which returns the best score or evaluation result that is output from the logic program.

In this paper, we apply the proposed tool to optimize the parameters of the *throughput estimation model* for the wireless communication link in a wireless local-area network (WLAN) [3–5]. The accurate throughput estimation model plays an important role in designing efficient WLAN systems [6–8]. Our model first calculates the receiving signal strength at the receiver using the *Log distance path loss model* [9]. Then, it calculates the throughput using the *sigmoid function* originally. Each model/function has several parameters to be tuned, which are optimized by the tool. By comparing the throughput estimation errors between the manual tuning and the tool, the effectiveness of the tool is verified.

In [10], *Dakota* was presented as a software tool with similar functions. It has been developed by the Sundia National Research Laboratory in USA as open source software. The comparisons with our tool will be involved in future works.

The rest of this paper is organized as follows: Sects. 2 and 3 present the proposed parameter optimization tool and its adopted algorithm respectively. Section 4 shows the application to the throughput estimation model for WLANs. Finally, Sect. 5 remarks the conclusion of this paper with future works.

## 2 Parameter Optimization Tool

In this section, we present the proposed parameter optimization tool. The parameter optimization tool is independent from the logic program for generalization, running the program as its child process. The effectiveness of the parameter values to run the logic program is evaluated by the output score from the program given in the output file. This tool searches the parameter values that maximize or minimize the output score. Currently, it is implemented by *Java* as a desktop application. In future works, it will be implemented as a Web application.

A user of the tool is required to prepare the following five files.

**1. Parameter Specification File**

The parameter specification file describes the condition how to change the value of each parameter during the search process. “parameter.csv” must be used as the file name. Each line in this file reflects the specification for one parameter, and must describe in the order of “parameter name”, “initial value”, “lower limit”, “upper limit”, and “change step”. The *change step* indicates one step

change size of the parameter value when the tool changes it to generate a new value. For example, when the *parameter name* is $x$, the *initial value* is 50, the *lower limit* is 0, the *upper limit* is 100, and the *change step* is 5, $x$ is initially assigned 50 and then, it can be changed to 0, 5, ..., 100 during the optimization process.

**2. Logic Program File**
The logic program file is a binary code file to run the logic, which can be executed through the command line. Any name is possible for this file. This file must satisfy the following two conditions:

(1) When the program is executed, it receives the path for the parameter file in the argument and applies the parameter values in the file to the algorithm/logic.
(2) When the program is completed, it outputs the score as the evaluation value in the text file "result.txt".

With (1), the logic program can read the parameter values that are generated by the tool. With (2), the tool can read the score that is calculated in the logic program. For example, the parameter file for the previous example with three parameters is described as follows when their initial ones are used:

**3. Sample Input Data File**
The sample input data file contains the input data set to the logic program such that the result of the logic program is evaluated and used to optimize the parameter values in the tool. To improve the accuracy of the obtained parameter values, multimodal sample input data sets should be collected and adopted in the tool.

**4. Score Output File**
The score output file contains the score from the logic program to evaluate the current parameter values. The score can be given by the difference between measured and estimated throughputs in the throughput estimation model, the cost function for an optimization problem, and the S/N ratio for a signal processing filter.

**5. Script File for Execution**
The script file describes the sequence of the commands to execute the logic program. The file name must be "run.sh". It describes the paths to the input files for the logic program. By modifying this script file, the user can change the name and the arguments for the logic program, and may run multiple logic programs sequentially to obtain one score. In the following example, line 10 executes the Java logic program with three paths to the input files. When the logic program is executed with multiple sample input data files continuously, the array to describe these files should be prepared and the loop procedure be adopted.

Example of "run.sh".

```
01: #!/bin/bash
02: DIR=$(cd $(dirname $0); pwd)
03: PARAMETER_FILE=$1
04: cd "$DIR"
05: #folder for sample input data files
06: INPUT_DIR="./file/3f_ap1/"
07: #measurement data file
08: MEASUREMENT_FILE="./file/evaluate/
  indoor_3f_ap1.csv"
09: #execute the logic program
10: java -jar ThroughputEstimation.jar
  ${INPUT_DIR} ${MEASUREMENT_FILE}
  $PARAMETER_FILE
11: #move ‘‘result.txt" to folder of parameter file
12: mv result.txt ‘dirname $PARAMETER_FILE‘
```

The processing flow of the proposed tool is shown as follows:

(1) The parameter optimization tool ($T$) generates the initial parameter file by copying the initial values in the parameter specification file.
(2) $T$ executes the script file using the current parameter file.
   (a) The logic program ($L$) reads one sample input data file.
   (b) $L$ computes the algorithm/logic.
   (c) $L$ writes the score in the score output file.
(3) $T$ reads the score from the score output file.
(4) When the termination condition is satisfied, $T$ goes to (5), otherwise goes to (6).
(5) $T$ changes the parameter file based on the algorithm in the next section, and goes to (2).
(6) $T$ selects the parameter values with the best score and outputs it.

## 3 Parameter Optimization Algorithm

In this section, we describe the parameter optimization algorithm in the tool [1,2]. The following symbols are used in the algorithm:

- $P$: the set of the $n$ parameters for the algorithm/logic in the logic program whose values should be optimized.
- $p_i$: the $i$th parameter in $P$ ($1 \leq i \leq n$).
- $\Delta p_i$: the change step for $p_i$.
- $t_i$: the tabu period for $p_i$ in the tabu table.
- $S(P)$: the score of the algorithm/logic using $P$.
- $P_{best}$: the best set of the parameters.
- $S(P_{best})$: the score of the algorithm/logic where $P_{best}$ is used.
- $L$: the log or cashe of generated parameter values and their scores.

The algorithm procedure that minimizes the score is described as follows:

(1) Clear the generated parameter log $L$.
(2) Set the initial value in the parameter file for any $p_i$ in $P$, and set 0 for any tabu period $t_i$, and set a large value for $S(P_{best})$.
(3) Generate the neighborhood parameter value sets for $P$ by:
   (a) Randomly select one parameter $p_i$ for $t_i = 0$.
   (b) Calculate parameter values of ${p_i}^-$ and ${p_i}^+$ by:

$$\begin{aligned} {p_i}^- &= p_i - \Delta p_i, \text{if } p_i > \text{lower limit}, \\ {p_i}^+ &= p_i + \Delta p_i, \text{if } p_i < \text{upper limit}. \end{aligned} \tag{1}$$

   (c) Generate the neighborhood parameter value sets $P^-$ and $P^+$ by replacing $p_i$ by ${p_i}^-$ or ${p_i}^+$:

$$\begin{aligned} P^- &= \{p_1, p_2, \ldots, {p_i}^-, \ldots, p_n\} \\ P^+ &= \{p_1, p_2, \ldots, {p_i}^+, \ldots, p_n\} \end{aligned}$$

(4) When $P$ ($P^-$, $P^+$) exists in $L$, obtain $S(P)$ ($S(P^-)$, $S(P^+)$) from $L$. Otherwise, execute the logic program using $P$ ($P^-$, $P^+$) to obtain $S(P)$ ($S(P^-)$, $S(P^+)$), and write $P$ and $S(P)$ ($P^-$ and $S(P^-)$, $P^+$ and $S(P^+)$) into $L$.
(5) Compare $S(P)$, $S(P^-)$, and $S(P^+)$, and select the parameter value set that has the largest one among them.
(6) Update the tabu period by:
   (a) Decrement $t_i$ by $-1$ if $t_i > 0$.
   (b) Set the given constant tabu period $TB$ for $t_i$ if $S(P)$ is largest at (5) and $p_i$ is selected at (3)(a).
(7) When $S(P)$ is continuously largest at (5) for the given constant times, go to (8). Otherwise, go to (3).
(8) When the hill-climbing procedure in (9) is applied for the given constant times $HT$, terminate the algorithm and output $P_{best}$. Otherwise, go to (9).
(9) Apply the hill-climbing procedure:
   (a) If $S(P) < S(P_{best})$, update $P_{best}$ and $S(P_{best})$ by $P$ and $S(P)$.
   (a) Reset $P$ by $P_{best}$.
   (c) Randomly select $p_i$ in $P$, and randomly change the value of $p_i$ within its range.
   (d) Go to (3).

## 4 Application to Throughput Estimation Model for WLAN

In this section, we reveal the application results of the proposed tool to the throughput estimation model for WLAN in [3,4] with the model extension to improve the accuracy.

### 4.1 Overview of Throughput Estimation Model

This model estimates the throughput or transmission speed of a wireless communication link between a source node and a destination node in WLAN. First, this model estimates the receiving signal strength at the destination node using the *Log distance path loss model* by considering the distance and the obstacles between the end nodes. Next, it estimates the throughput using the *sigmoid function* from the receiving signal strength. Each function has several parameters that can affect the estimation accuracy. In [4], three network fields composed of multiple small rooms are considered, and throughputs between APs and hosts in different locations were measured using *Iperf* [11] and estimated by the model, where the parameter values for the model were tuned manually. For now, we discuss the results regarding that the source node is an AP and the destination node is a host.

The results in [4] show that prominent differences exist in specific links between estimated throughputs and measured ones. As the reason, they estimated that the model only considers the *direct signal* along the *line of sight (LOS)* at the host. It does not consider the *multipath effect* due to the *indirect signal* reflected at the walls, even if the indirect signal is stronger than the direct signal. The direct signal becomes exceedingly weak when it passes several walls or obstacles along the path. If the indirect signal passes only a few walls, the receiving signal strength of the host of this indirect signal can be much larger and the estimated throughput can be increased.

Therefore, to consider such indirect signals, we propose the extension of the throughput estimation model. For simplicity, we assume that for each host in the field, there exists a *reflecting point* on the wall in the same room such that the signal strength of the AP is the largest, and the indirect signal reaches the host through this reflecting point after the signal is attenuated there.

### 4.2 Input and Output Files

The input files to the logic program for the throughput estimation model are as follows:

- Network field file: it describes the set of the two coordinates of the end points on each wall and the wall type.
- Device coordinate file: it describes the set of the coordinates for one AP and plural hosts.
- Reflecting point file: it describes the set of the coordinates of the reflecting points for the hosts.
- Parameter file: it describes the parameter values of the model.
- Sample input data file: it describes the measured signal strengths and throughput at the hosts.

The summation of the absolute value of the difference between each measured and estimated throughput is output as the score for the current parameter values in the score output file.

### 4.3 Throughput Estimation Procedure

The computation procedure of the throughput estimation model including the extension is as follows:

(1) Calculate the Euclid distance $d$ $(m)$ for each link (AP/host pair) by:

$$d = \sqrt{(AP_x - H_x)^2 + (AP_y - H_y)^2} \tag{2}$$

where $AP_x$ and $AP_y$ represent the $x$ and $y$ coordinates for the AP, and $H_x$ and $H_y$ represent the $x$ and $y$ coordinates for the host.

(2) Find the walls intersecting with the link. Here, the link and the wall are regarded as line segments where their intersection is judged by considering the intersection between the two line segments.

(3) Estimate the receiving signal strength at the host using the Log distance path loss mode and consider the indirect signal:

(a) Calculate the receiving signal strength $P_{di}$ at the host through the direct signal:

$$P_{dir} = P_1 - 10\alpha \log_{10} d - \sum_k n_k W_k \tag{3}$$

where $P_{dir}$ represents the receiving signal strength $(dBm)$ at the host with the distance $d$ $(m)$ from the AP, $P_1$ does the signal strength $(dBm)$ at the host with $1m$ distance from the source node, $\alpha$ does the attenuation factor that should be optimized, $n_k$ does the number of walls with type $k$, and $W_k$ does the signal attenuation factor $(dBm)$ of the type $k$ wall.

(b) Calculate the receiving signal strength through the indirect signal:

(1) Calculate the receiving signal strength $P_{ref}$ $(dBm)$ at the reflecting point for each host using (4).

(2) Calculate the receiving signal strength $P_{ind}$ $(dBm)$ at the host by the indirect signal, where the receiving signal strength at (1) is used as the transmitting signal strength from the reflecting point using (5).

$$P_{ref} = P_1 - 10\alpha \log_{10} r - \sum_k n_k W_k \tag{4}$$

$$P_{ind} = P_r - 10\alpha \log_{10} t - W_{ref} \tag{5}$$

where $r$ $(m)$ represents the distance between the AP and the reflecting point, $t$ $(m)$ does the distance between the reflecting point and the host, and $W_{ref}$ $(dBm)$ does the attenuation factor at the reflecting point.

(c) Select the larger one between $P_{dir}$ and $P_{ind}$ for the receiving signal strength $P_h$ $(dBm)$ at the host.

(4) Calculate the estimated throughput $S_h$ $(Mbps)$ using the sigmoid function in (6).

$$S_h = \frac{a}{1 + exp(\frac{P_h - b}{c})} \tag{6}$$

where $a$, $b$, and $c$ are constant coefficients.

### 4.4 Application of Tool

By applying the proposed tool, the values of the parameters for the throughput estimation model, namely, $P_1$, $\alpha$, $W_k$, and $W_{ref}$ in (3), (5), and (4), and $a$, $b$, and $c$ in (6) are optimized. $W_k$ is prepared for each wall type. In this paper, the corridor wall ($k = 1$), the partition wall ($k = 2$), the intervening wall ($k = 3$), the glass wall ($k = 4$), and the elevator wall ($k = 5$) are considered. For the parameters in the parameter optimization algorithm, $TB = 16$ and $HT = 10$ are adopted. For the sample input data files, the measurement results in [4] are used in the application of the tool.

The three network fields in [4] are considered. Here, in field#1, one additional AP is allocated between the hosts 307-3 and Ref-3 and one AP is at the center in D307, in addition to the AP in [4]. In field#2, two additional APs are also allocated at the similar positions as in field#1. The reflecting point for each host is manually selected. Because field#1 and field#2 are located in the same building on different floors, all the sample input data sets measured with six

**Table 1.** Throughput estimation errors.

| AP | method | model | ave. | max. | min. | var. |
|---|---|---|---|---|---|---|
| field#1 AP1 | manual | conv. | 9.42 | 18.7 | 0.62 | 38.7 |
| | tool | conv. | 5.15 | 20.74 | 0.01 | 46.2 |
| | tool | extend. | 5.81 | 22.6 | 0 | 56.0 |
| field#1 AP2 | manual | conv. | 11.4 | 26.0 | 0.58 | 65.9 |
| | tool | conv. | 9.81 | 20.9 | 0.97 | 48.5 |
| | tool | extend. | 9.20 | 19.3 | 0 | 51.6 |
| field#1 AP3 | manual | conv. | 17.3 | 34.4 | 0.44 | 89.0 |
| | tool | conv. | 16.5 | 47.3 | 5.22 | 146.9 |
| | tool | extend. | 16.2 | 46.6 | 3.24 | 146.5 |
| field#2 AP1 | manual | conv. | 19.1 | 34.6 | 5.07 | 90.8 |
| | tool | conv. | 13.2 | 31.5 | 1.19 | 62.6 |
| | tool | extend. | 12.6 | 31.4 | 0.51 | 64.0 |
| field#2 AP2 | manual | conv. | 18.7 | 31.8 | 4.96 | 64.6 |
| | tool | conv. | 12.6 | 21.2 | 3.08 | 24.6 |
| | tool | extend. | 12.3 | 21.0 | 2.10 | 33.8 |
| field#2 AP3 | manual | conv. | 15.4 | 33.9 | 0.46 | 80.3 |
| | tool | conv. | 17.5 | 41.0 | 0.65 | 149.7 |
| | tool | extend. | 18.1 | 42.0 | 0.81 | 148.4 |
| field#3 AP1 | manual | conv. | 13.2 | 38.2 | 0.83 | 116.7 |
| | tool | conv. | 6.75 | 1.45 | 0 | 28.5 |
| | tool | extend. | 5.05 | 15.82 | 0 | 25.8 |

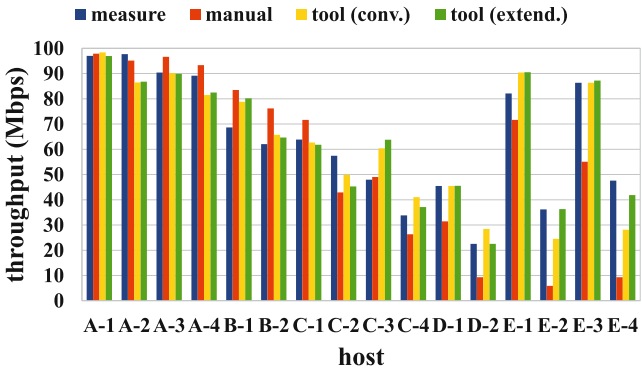


**Fig. 1.** Throughputs for AP1 in field#3.

APs and 28 hosts in the two fields are used together to select the optimal values of the parameters.

Then, we compute the throughput estimation model using the optimized parameter values. Table 1 summarizes the average, maximum, and minimum scores or estimation errors and the variances for each AP in the three fields. For reference, Fig. 1 shows the measured and estimated throughputs for the hosts with AP1 in field#3. Table 1 indicates that the accuracy of the throughput estimations of the model is generally improved by the tool, although it is degraded for certain hosts. The analysis of the reason and the provision will be explored in future works. The accuracy for the two hosts, E-2 and E-4, with AP1 in field#3, is greatly improved by the extended model. These results have verified the contribution of the proposed tool and model extension in this paper.

## 5 Conclusion

This paper presented a versatile parameter optimization tool to find optimal values of parameters for various algorithms or logics to solve mathematical problems. The effectiveness of the tool was confirmed through applications to the throughput estimation model for wireless local-area networks, where the extension of the model was also presented to enhance the accuracy. In future works, the tool will be applied to different algorithms/logics for verifications, and the user interface will be improved including the implementation of the Web application.

## References


1. Sekioka, T., Funabiki, N., Higashino, T.: A proposal of an improved function synthesis algorithm using genetic programming. IEICE Trans. D1 **J83–D–I**(4), 407–417 (2000)
2. Sekioka, T., Yokogawa, Y., Funabiki, N., Higashino, T., Yamada, T., Mori, E.: A proposal of a lip contour approximation method using the function synthesis. IEICE Trans. D2 **J84–D–II**(3), 459–470 (2001)
3. Mamun, M.S.A., Islam, M.E., Funabiki, N., Kuribayashi, M., Lai, I.-W.: An active access-point configuration algorithm for elastic wireless local-area network system using heterogeneous devices. Int. J. Network. Comput. **6**(2), 395–419 (2016)
4. Lwin, K.S., Funabiki, N., Zaw, K.K., Mamun, M.S.A., Kuribayashi, M.: A minimax approach for access-point setup optimization using throughput measurements in IEEE802.11n wireless networks. In: Proceedings of the CANDAR 2016 (2016)
5. Funabiki, N., Lwin, K.S., Kuribayashi, M., Lai, I.W.: Throughput measurements for access-point installation optimization in IEEE802.11n wireless networks. In: Proceedings of the IEEE ICCE-TW, pp. 218–219 (2016)
6. Gibney, A.M., Klepal, M., Pesch, D.: A wireless local area network modeling tool for scalable indoor access point placement optimization. In: Proceedings of the SpringSim, pp. 163–170 (2010)
7. Farkas, K., Huszák, Á., Gódor, G.: Optimization of Wi-Fi access point placement for indoor localization. Network. Commun. **1**(1), 28–33 (2013)
8. Safna, R.F., Manoshantha, E.J.N., Suraweera, S.A.T.S., Dissanayake, M.B.: Optimization of wireless pathloss model JTC for access point placement in wireless local area network. In: Proceedings of the RSEA 2015, pp. 235–238 (2015)
9. Faria, D.B.: Modeling signal attenuation in IEEE 802.11 wireless LANs. Technical Report, TR-Kpp. 06-0118, Stanford University (2005)
10. Dakota. https://dakota.sandia.gov/
11. ACD.net, Iperf Speed Testing. http://support.acd.net/wiki/index.php?title=Iperf Speed Testing