Identifying optimum Artificial Bee Colony (ABC) algorithm's parameters for scheduling the manufacture and assembly of complex products

Primpika Pansuwan
Faculty of Science, Naresuan University, Amphur Muang, Pitsanulok, 65000 Thailand
Email: pupongp@yahoo.com or pupongp@nu.ac.th

Niyada Rukwong
Faculty of Science, Naresuan University, Amphur Muang, Pitsanulok, 65000 Thailand.
Email: pupongp@yahoo.com or pupongp@nu.ac.th

Pupong Pongcharoen
Faculty of Engineering, Naresuan University, Amphur Muang, Pitsanulok, 65000 Thailand.
Email: pupongp@yahoo.com or pupongp@nu.ac.th

Abstract — **Production scheduling in multiple-stage multiple-machine multiple-product environment is a NP hard problem usually faced by make/engineer-to-order companies engaged in capital goods Industry. Feasible schedules must correctly sequence the operations required to manufacture components and also satisfy assembly precedence relationships. This paper presents the development of Artificial Bee Colony algorithm for solving the scheduling problem. Based on Just in time philosophy, the proposed algorithm was designed to minimise the combination of earliness and tardiness penalties cost. The computational experiment was conducted using data obtained from a collaborating company that manufactures complex capital goods. The aim was to investigate the influence of parameter configuration on the algorithm performance. The analysis of variance on the experimental results indicated that the performance can be improved dramatically after adopting the optimum parameter setting.**

Keywords - Production scheduling, Make/engineer to order, Complex product, Computational intelligence, Artificial Bee Colony algorithm.

I. INTRODUCTION

Sequencing determines the order of tasks based upon operation and assembly precedence. Scheduling is defined as "the allocation of resources over time to perform a collection of tasks" [1]. A schedule specifies sequence and timing, normally expressed in a set of start and due times. Scheduling is a combinatorial optimisation problem and classified as Non-deterministic polynomial (NP) hard problem [2], which means that the amount of computation required to find solution increase exponentially with problem size.

There have been a number of research articles related to classical job/flow shop scheduling problems that generally consist of a set of independent jobs or tasks to be performed on machines. This is a single stage scheduling [3], which means that there are no precedence constraints among operation and assembly requirements [4]. Scheduling is important because companies seek to deliver products in minimum time and simultaneously achieve high resource utilisation.

Various assumptions have been made in order to simplify, formulate and solve scheduling problems. The most common assumptions can be summarised as follows [5]: a successor operation is performed immediately after its predecessor is finished, providing that the machine is available; each machine can handle only one operation at a time; each operation can only be performed on one machine at a time; No interruption of operations; No rework; and Setup and transfer times are of zero or uniform duration and tasks are independent.

Multi-stage multi-machine production scheduling with multi-product especially occurred in capital goods industry is difficult for several reasons. Demand is highly variable and uncertain. The industries usually produce capital products (e.g. steam turbine generators, power station boilers and transformer), each of which required a large number of operations on machines with high capital and operating cost. There are many dependency relationships, e.g., due to product structure relationships. There are also multiple finite capacity resource constraints and performance objectives. It requires different control approaches for machine with high and low utilisation [6, 7]. Reeja and Rajendran [8] highlighted the lack of production scheduling that took account of assembly relationships and constraints.

The objectives of this paper were to: i) present the development of the Artificial Bee Colony algorithm for solving multiple-stage multiple-machine multiple-product scheduling problems; ii) investigate the appropriate parameter setting of the proposed method using statistical tools for experimental design and analysis; and iii) compare the algorithm's performance of the with and without the optimised parameter setting.

The remaining sections firstly present a brief introduction on scheduling problem in multiple-stage multiple-machine multiple-product environment followed by the process of the Artificial Bee Colony (ABC) algorithm and it pseudo code for scheduling the manufacture and assembly of complex products, which describes in section 3. Section 4 presents the experimental design and provides a statistical analysis on the experimental results. These are followed by the conclusions in section 5.

II. Production scheduling problem

Multi-stage, multi-machine, multi-product scheduling (MMMS) is the allocation of resources over time to perform a collection of operations required by components, which are subsequently sub-assembled and assembled into products in accordance with the product structure. The number of levels of product structure indicates the number of stages of assembly precedence relationships. The leaf nodes within the product structure represent components. Since some components may require a number of operations to be performed on various machines, the sequence of operations for each component must consider the operation precedence constraint inherent its manufacturing process.

Previous research relating to the multi-stage multi-machine multi-product scheduling (MMMS) problem has been focused on the application of metaheuristics e.g. Ant Colony Optimisation [9] and Shuffled Frog Leaping [10]. However those research works have not provided an exact mathematical formulation of the problem [11, 12]. However, Chen and Ji [13] applied a mixed integer programming method to solve a MMMS problem. Due to the limitation of their proposed method, the numerical experiments were based on a problem that consisted of 2 final products with a requirement for 21 operations to manufacture 21 items (excluding final products). In order to formulate the multi-stage, multi-machine, multi-product scheduling (MMMS) model, the following notation is introduced for using in the model.

Nottions:

i	operation i^{th} ($i = 1, \ldots, O$)
j	part or component j^{th} ($j = 1, \ldots, C$)
k	final product k^{th} ($k = 1, \ldots, P$)
m	machine m^{th} ($m = 1, \ldots, M$)
E_k	earliness duration of product k^{th}
E_{jk}	earliness duration of component j^{th} in product k^{th}
T_k	tardiness duration of product k^{th}
$X_{ijkabcm}$	1 if operation i^{th} for component j^{th} in product k^{th} precedes operation a^{th} for component b^{th} in product c^{th} on machine m^{th}; and 0 otherwise
R_m	ready time of machine m^{th}
C_k	completion time of product k^{th}
C_{jk}	completion time of component j^{th} in product k^{th}
D_k	due date of product k^{th}
D_{jk}	due date of component j^{th} in product k^{th}
SU_{ijkm}	setup time of operation i^{th} for component j^{th} in product k^{th} on machine m^{th}
ST_{ijkm}	start time of operation i^{th} for component j^{th} in product k^{th} on machine m^{th}
PT_{ijkm}	processing time of operation i^{th} for component j^{th} in product k^{th} on machine m^{th}
FT_{ijkm}	finishing time of operation i^{th} for component j^{th} in product k^{th} on machine m^{th}
TT_{ijkm}	transfer time of operation i^{th} for component j^{th} in product k^{th} on machine m^{th}
Pe	earliness penalty (currency unit per day)
Pt	tardiness penalty (currency unit per day)
$S(x)$	set of child items of item x
Sh	working hour per shift (minutes per shift)

The scheduling objective is based on the Just in Time philosophy aiming to minimise the combination of earliness penalties for components and final products and tardiness penalties of final products. The mathematical model for the MMMS problem is as follows:

$$\text{Penalty cost} = \sum_{j=1}^{C}\sum_{k=1}^{P} Pe(E_{jk}) + \sum_{k=1}^{P} Pe(E_k) + \sum_{k=1}^{P} Pt(T_k) \qquad (1)$$

Subject to:

$$ST_{ijkm} \geq R_m \qquad \forall i,j,k,m \qquad (2)$$
$$FT_{ijkm} = ST_{ijkm} + SU_{ijkm} + PT_{ijkm} + TT_{ijkm} \qquad \forall i,j,k,m \qquad (3)$$
$$C_{jk} \geq FT_{ijkm} \qquad \forall i,j,k,m \qquad (4)$$
$$E_{jk} = (D_{jk} - C_{jk})/Sh \qquad \forall j,k \qquad (5)$$
$$E_k = (D_k - C_k)/Sh \qquad \forall k \qquad (6)$$
$$T_k = (C_k - D_k)/Sh \qquad \forall k \qquad (7)$$
$$ST_{ihkm} - ST_{ijkm} \geq SU_{ijkm} + PT_{ijkm} + TT_{ijkm} \qquad \forall i,k,m,j \in S(h) \qquad (8)$$
$$ST_{gjkm} - ST_{ijkm} \geq SU_{ijkm} + PT_{ijkm} + TT_{ijkm} \qquad \forall j,k,m,g=i+1 \qquad (9)$$
$$X_{ijkabcm} + X_{abcijkm} = 1 \qquad \forall a,b,c,i,j,k,m \qquad (10)$$
$$X_{ijkabcm} \in (0, 1) \qquad \forall a,b,c,i,j,k,m \qquad (11)$$
$$E_{jk}, E_k, T_k \geq 0 \qquad \forall j,k \qquad (12)$$
$$ST_{ijkm}, R_m \geq 0 \qquad \forall i,j,k,m \qquad (13)$$
$$FT_{ijkm}, ST_{ijkm}, SU_{ijkm}, PT_{ijkm}, TT_{ijkm} \geq 0 \qquad \forall i,j,k,m \qquad (14)$$

This mathematical model has been derived to describe the MMMS problem considered in previous research [14]. The objective function (1) contains three parts: i) an earliness penalty for components; ii) an earliness penalty for final products; and iii) a tardiness penalty for final products. Constraint (2) ensures that the start time of all operations is not earlier than the time the machines are ready. Constraint (3) makes sure that the finishing time for each operation is determined by the start, setup, machining and transfer times. Constraint (4) ensures that the completion time for each component is not before the finishing time. The number of days earliness and tardiness for components and final products are defined by constraints (5), (6) and (7). Part precedence constraints relating to the product structure are defined by constraint (8). For example, if part j is child item of part h, then the start time of the parent part (ST_{ihkm}) minus the start time of the child item (ST_{ijkm}) should be greater or equal to the sum of setup, processing and transfer times for the child item. Constraint (9) makes sure that the operation precedence constraints within a component are satisfied. For example, if a successor operation (g) is performed after its predecessor operation (i) is finished, the start time of the successor operation (ST_{gjkm}) minus the start time of the predecessor operation (ST_{ijkm}) should be greater or equal to the sum of setup, processing and transfer times of the predecessor operation. Constraint (10) ensures that only one operation can be performed on a machine using the decision variables defined by constraints in (11). Constraints (12)-(14) guarantee non-negative values for those defined variables.

III. Artificial bee colony (ABC) algorithm

Since nature is always a source of inspiration, there has been increasing interests in development of computational models or methods that iteratively conduct stochastic search process inspired by natural intelligence. Many optimisation algorithms have been designed and developed by adopting a form of biological-based swarm intelligence including Artificial Bee Colony (ABC) Algorithm.

Honey Bee is a good example of well known social insects with self organisation and division of labour for food collection through information sharing between employed and unemployed foragers. Employed bees are associated with food sources, which they are currently exploiting or are employed at. Whilst unemployed bees are associated with establishing new food sources either by searching the environment surrounding the nest (called scouts) or waiting in the nest for sharing information related to food sources shared by employed bee (so called onlookers). The foraging behaviours lead to the model development of collective intelligence of honeybee swarms [15]. The main steps of the Artificial Bee Colony (ABC) algorithm including initialise population, place the employed bees on their food sources, place the onlooker bees on the food sources depending on their nectar amounts, send the scouts to search area for discovering new food sources and finally update the best food source found so far. The food searching process is repeated until termination criteria are satisfied [16].

In the ABC algorithm, there are three control parameters including i) the number of food sources, which is equal to the population size (S) of employed or onlooker bees; ii) the predefined value of limit (L) for unimproved loop in the case that if a position cannot be improved then food source is assumed to be abandoned; and finally iii) the maximum loop for searching food (M). These control parameters play an important role on the performance of the ABC algorithms. Enhancing the algorithm's performance can be basically accomplished by the use of the appropriate parameter setting, which can be systematically investigated and statistical identified via the experimental design and analysis [17, 18]. Due to the nature and complexity of the problem domains, it has been suggested that the appropriate parameter setting can be varied across problem sizes and/or problem domains. Most research work related to the application of nature-inspired algorithms e.g. Genetic Algorithm has set it parameters and operators in an ad hoc fashion [19].

The pseudo code of ABC algorithm applied to solve the production scheduling problem is shown in figure 1. The ABC based scheduling program was coded in modular style using a general purpose programming language called TCL/TK [20]. The scheduling program developed can be categorised into three phases: i) input phase, in which product's information and its manufacturing data including operation no., machining time, part code, product structure and resource were uploaded into the program; ii) scheduling phase, where the proposed ABC algorithms were used to generate and evaluate schedules constrained by precedence relationships and finite resource capacity; and iii) output phase including information on the best production schedule found and its penalty cost. Graphic user interface (GUI) was considered during the development of the program to allow users to manipulate data, set parameters and choose outputs from the program.

```
Define population size ($S$), loop ($M$) and unimproved loop ($L$)
Initial the population of schedules ($X_i$); $i = 1, 2, 3, \ldots, S$.
Evaluate the population
Set $loop = 1$
Repeat
    Produce new schedules ($V_i$) as the employed bees and
    evaluate them
    Apply greedy selection process for the employed bees
    Calculate the probability value ($P_i$) for schedules ($X_i$)
    Produce $V_i$ as the onlookers from $X_i$ selected depending on $P_i$
    and evaluate them
    Apply greedy selection process for the onlookers
    Determine the abandoned solution for the scout according to
    $L$, if exists, replace it with a new randomly produced schedule
    Update the best achieved so far solution
    $loop = loop + 1$
Until $loop = M$
```

Figure 1. Pseudo code of the ABC algorithm adopted from [16].

In this work, scheduling data including resources profile, products' information, manufacturing and operational data as well as customer due date obtained from a collaborating company currently engaged in make/engineer to order capital good industry were experimented as a case study. The considered problem consisted of two products, each of which has four levels of product structure. To manufacture both products, it involves a combined requirement of 118 machining operations and 17 assembly operations to be performed on 17 non-identical machines (resources). A notebook computers with Core2 Duo 1.80 GHz CPU and 2.5 GB DDRII RAM was used to determine the simulation time required to execute a computational run.

IV. Experimental design and analysis

This section presents the design and analysis on the computational experiment conducted using the ABC algorithm to solve production scheduling problem obtained from a collaborating company engaged in capital goods industry. The experiment was aimed to systematically investigate the appropriate parameter setting of the ABC algorithm via the statistical design and analysis. Full factorial experimental design (3^k) shown in table I was adopted in this work.

TABLE I. Experimental factors and its levels.

Factors	Levels	Uncoded Values		
		Low(-1)	***Medium(0)***	***High(1)***
SM	3	25*100	50*50	100*25
L	3	10	30	50

From table I, there are two factors, each of which is considered at three levels. The former factor is the combination of the population sizes (*S*) and the maximum loop (*M*) for searching solutions. This combination of both parameters plays an important role on the amount of search in the solution space conducted by the ABC algorithm. Higher values of both parameters increase the probability of finding the best solutions but require longer computational time. If there is no timing constraint, the values of both parameters should be defined as high as possible. However, time is always a constraint. In this work, the amount of search (a combination of *SM*) for the instants problem is predefined at 2,500. The latter factor is the predefined value of limit (*L*) for unimproved loop in the case that if a position cannot be improved then food source is assumed to be abandoned. Solving non-linear mathematical functions [16], the limitation has been set to the number of food source (*S*) multiplied by the dimension of the problem considered. Due to the production scheduling problem considered in this work, there is no dimension related. The limitation was therefore defined as percentage according to the maximum loop (*M*). The values of the parameter were investigated between 10-50% of the maximum loop (*M*).

The full factorial experimental (3^2) design [21] was carried out with five replications using different random seed numbers. The computational results obtained from 45 (3^2*5) runs were analysed using a general linear model form of analysis of variance (ANOVA). Table II shows ANOVA table consisting of Source of Variation, Degrees of Freedom (*DF*), Sum of Square (*SS*), Mean Square (*MS*), *F* and *P* values. A factor with value of $P \leq 0.01$ was considered statistically significant with 99% confidence interval.

TABLE II. ANOVA ON THE ABC'S PARAMETERS.

Source	*DF*	*SS*	*MS*	*F*	*P*
SM	2	290711111	145355556	31.06	0.000
L	2	23644444	11822222	2.53	0.094
*SM**L*	2	156788889	39197222	8.37	0.000
Error	36	168500000	4680556		
Total	44				

From ANOVA table, it can be seen that all ABC algorithm's parameters (*SM* and *L*) were statistically significant either in form of main effect and/or interaction with 99% confidence interval. The combination of the population sizes and the maximum loop (*SM*) for searching solutions was found significant as main effect. The main effect plots shown in figure 2 indicated that the best schedules were obtained from the ABC algorithm when the *SM* was set at 50*50. Another parameter, the value of limit (*L*), was almost significant in main effect but was significant in the interaction effect with 99% confidence interval.

The interaction effect plots of the ABC algorithm shown in figure 3 indicated that if the *SM* parameter was set at 50*50 then the *L* parameter can be set either value between 10-50% of the maximum loop (*M*). However, if the *SM* parameter was set at 100*25 then the *L* parameter should be set between 30-50%. But if the *SM* parameter was set at 25*100 then the *L* parameter should be set at 10%.

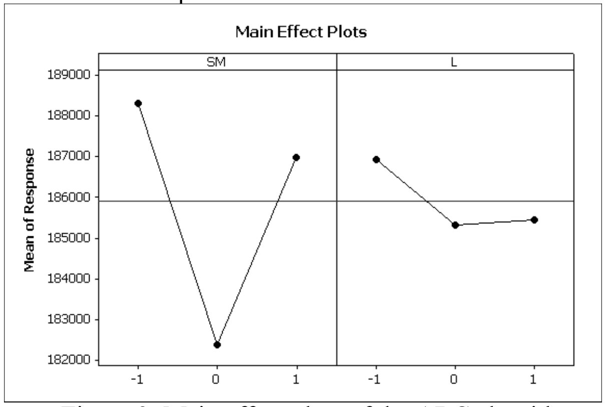

Figure 2. Main effect plots of the ABC algorithm.

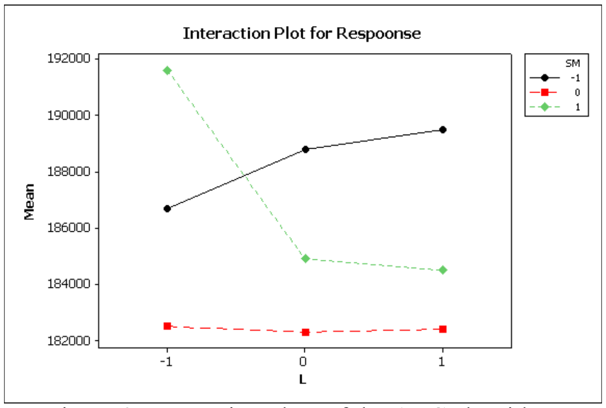

Figure 3. Interaction plots of the ABC algorithm.

According to the statistical analysis on the average and standard deviation of the computational results obtained from different setting of the ABC parameter configurations (*SM*+*L*) shown in figure 4, it can be seen that the ABC algorithm parameters (*SM* and *L*) were recommended at 50*50 and 10-50%, respectively.

As far as the computational time is concerned, the execution time required for each experimental run was recorded. It was found that the average execution time taken by the ABC algorithm to find the best so far schedules was about 3.33 minutes.

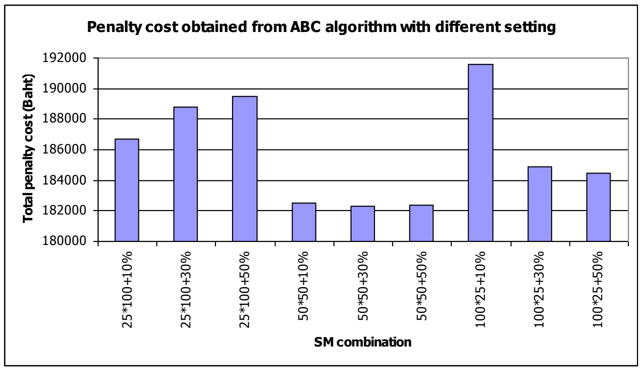

Figure 4. Interaction plots of the ABC algorithm.

V. Conclusions

Multi-stage multi-machine multi-product scheduling (MMMS) problem is classified as a NP hard problem, which means that the amount of computation required to find solution increase exponentially with problem size. Feasible schedules must correctly sequence the operations required to manufacture components and also satisfy assembly precedence relationships. This paper presents the development of Artificial Bee Colony (ABC) algorithm for solving the MMMS problem. The proposed algorithm was designed to minimise the combination of earliness and tardiness penalties cost. The computational experiment based on industrial data obtained from a collaborating make/engineer-to-order company manufacturing complex capital goods was aimed to investigate the influence of parameter configuration on the algorithm performance. It was found that the algorithm's performance can be improved significantly after adopting the optimum parameter setting identified through statistical design and analysis.

VI. Acknowledgement

The corresponding author would like to acknowledge the Naresuan University since this work was part of the research project supported by the Naresuan University Research Fund under the grant number EN-AR-053/2552.

VII. Refferences

[1] K.R. Baker, *Introduction to Sequencing and Scheduling*. New York: Wiley and Sons, 1974.

[2] J.R. King and A.S. Spackis, "Scheduling: bibliography and review," *Int. J. of Physical Distribution and Materials Management*, vol. 10, pp. 105-132, 1980.

[3] Y. He and C.W. Hui, "Genetic algorithm based on heuristic rules for high-constrained large-size single-stage multi-product scheduling with parallel units," *Chem. Engi. and Proc.*, vol. 46, pp. 1175-1191, 2007.

[4] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Scheduling Computer and Manufacturing Processes*. Berlin: Springer, 1996.

[5] P. Pongcharoen, A. Khadwilard, and C. Hicks, "A Genetic Algorithm with a New Repair Process for Solving Multi-stage, Multi-machine, Multi-product Scheduling Problems," *Indu. Engi. and Manag. Syst.*, vol. 7, pp. 204-213, 2008.

[6] C. Hicks and P. Pongcharoen, "Dispatching rules for production scheduling in the capital goods industry," *Int. J. of Prod. Res.*, vol. 104, pp. 154-163, 2006.

[7] C. Hicks and P. Pongcharoen, "Applying different control approaches for resources with high and low utilisation: a case study of the production of complex products with stochastic processing times," *Int. J. of Tech. Manag.*, vol. 48, pp. 202-218, 2009.

[8] M.K. Reeja and C. Rajendran, "Dispatching rules for scheduling in assembly job shops - Part I," *Int. J. of Prod. Res.*, vol. 38, pp. 2051-2066, 2000.

[9] A. Chainual, T. Lutuksin, and P. Pongcharoen, "Computer based scheduling tool for multi-product scheduling problems," *Int. J. of the Computer, the Internet and Management*, vol. 15, pp. 241-246, 2007.

[10] S. Chompooming and P. Pongcharoen, "Application of Shuffled Frog Leaping Algorithm for Production Scheduling of Complex Products," *SWU Engineering Journal*, Article in press, 2010.

[11] P. Pongcharoen, C. Hicks, and P.M. Braiden, "The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure," *Eur. J. of Oper. Res.*, vol. 152, pp. 215-225, 2004.

[12] P. Pongcharoen, C. Hicks, P.M. Braiden, and D.J. Stewardson, "Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products," *Int. J. of Prod. Res.*, vol. 78, pp. 311-322, 2002.

[13] K. Chen and P. Ji, "A mixed integer programming model for advanced planning and scheduling (APS)," *Eur. J. of Oper. Res.*, vol. 181, pp. 515-522, 2007.

[14] P. Pongcharoen, A. Chainual, A. Khadwilard, C. Kaweesirikon, and C. Hicks, "Comparison of mataheuristics for solving multi-product multi-stage multi-machine scheduling problems," presented at the 14th International Working Seminar on Production Economics, Innsbruck, 2008.

[15] V. Tereshko, "Reaction-diffusion model of a honeybee colony's foraging behaviour," *Parallel Problem Solving from Nature VI. Lecture Notes in Computer Science*, vol. 1917, pp. 807-816, 2000.

[16] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Appl. Math. and Comp.*, vol. 214, pp. 108-132, 2009.

[17] P. Pongcharoen, W. Chainate, and P. Thapatsuwan, "Exploration of genetic parameters and operators through travelling salesman problem," *ScienceAsia*, vol. 33, pp. 215-222, 2007.

[18] P. Pongcharoen, D.J. Stewardson, C. Hicks, and P.M. Braiden, "Applying designed experiments to optimize the performance of genetic algorithms used for scheduling complex products in the capital goods industry," *J. of Appl. Stat.*, vol. 28, pp. 441-455, 2001.

[19] H. Aytug, M. Knouja, and F.E. Vergara, "Use of genetic algorithms to solve production and operations management problems: a review," *Int. J. of Prod. Res.*, vol. 41, pp. 3955–4009, 2003.

[20] J.K. Ousterhout, *Tcl and the tk toolkit*. Massachusetts: Addison-Wesley, 1994.

[21] D.C.Montgomery, *Design and Analysis of Experiments*, 5 ed. New York: John Wiley & Sons, 2001.