J Optim Theory Appl (2009) 140: 213–231
DOI 10.1007/s10957-008-9444-9

Ordinal Optimization of G/G/1/K Polling Systems with k-Limited Service Discipline

S.-C. Horng · S.-Y. Lin

Published online: 19 December 2008

Abstract In this paper, we propose an ordinal optimization theory based algorithm to solve the optimization problem of G/G/1/K polling system with k-limited service discipline for a good enough solution using limited computation time. We assume that the arrival rates do not deteriorate visibly within a very short period. Our approach consists of two stages. In the first stage, we employ a typical genetic algorithm to select $N = 1024$ roughly good solutions from the huge discrete solution space Ω using an offline trained artificial neural network as a surrogate model for fitness evaluation. The second stage consists of several substages to select estimated good enough solutions from the previous N, and the solution obtained in the last substage is the good enough solution that we seek. Using numerous tests, we demonstrate: (i) the computational efficiency of our algorithm in the aspect that we can apply our algorithm in real-time based on the arrival rate assumption; (ii) the superiority of the good enough solution, which achieves drastic objective value reduction in comparison with other existing service disciplines. We provide a performance analysis for our algorithm based on the derived models. The results show that the good enough solution that we obtained is among the best $3.31 \times 10^{-6}\%$ in the solution space with probability 0.99.

Keywords Polling systems · k-Limited service disciplines · Stochastic simulation optimization · Ordinal optimization · Performance analysis

Communicated by Y.C. Ho.

This research work was supported in part by the National Science Council of Taiwan, ROC, Grant NSC95-2221-E-009-099-MY2.

S.-C. Horng
Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung, Taiwan, ROC
e-mail: schong@cyut.edu.tw

S.-Y. Lin (✉)
Department of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC
e-mail: sylin@cc.nctu.edu.tw

1 Introduction

The polling system plays an important role in the modeling and analysis of computer networks, communications, manufacturing systems, transportation systems, production system and inventory systems [1]. This system is a multiqueue served by a single cyclic server; its performance (for example, mean waiting time) is closely related with the queuing and service disciplines. In most of the existing literature [2, 3], first-in-first-out (FIFO) is usually employed as the queuing discipline. However, there are five typical service disciplines: exhaustive, gated, limited, k-limited and time-limited. In an exhaustive service discipline system, the server continues to serve all customers at a queue until it empties. In a gated service discipline system, the server continuously serves only those customers that are found at a queue when it is inspected [4]. In a limited service discipline system, at most one customer is served at a queue in a cycle. In a k-limited service discipline system, the server continues to serve k customers or until the queue is empty, whichever comes first. In a time-limited service discipline system [5], the server dwells certain amount of time at a queue even if it is empty, which is obviously inefficient.

Numerous analysis techniques had been developed to evaluate the mean waiting time of the M/M/1 polling system using exhaustive, gated, and limited service disciplines [6]. To relax the assumption on infinite buffer, Takagi also used exhaustive, gated, and limited service disciplines to analyze the M/G/1/K polling system [7]; Jung and Un [8] used the buffer occupancy method and the vacation results to derive the mean waiting time and blocking probabilities for the M/G/1/K polling system using exhaustive service discipline. However, the exhaustive, gated, and limited service disciplines are incapable of taking priority among queues to improve the system performance; thus, Borst *et al*. proposed a k-limited service discipline in [9] to find the optimal service limits in an M/G/l polling system. In general, the analysis of the performance of a polling system using k-limited service discipline is very difficult. Similar situation occurs to the time-limited service discipline, and the reason for its intended inefficiency as indicated previously is for the sake of obtaining an analytical result for the M/G/1 polling system.

From the above description, we see that the *queuing theory* based analytical results concerning the performance of a polling system were obtained at the expense of some restricting assumptions, for example the Poisson arrival process or infinite buffer and the apparent inefficiency in time-limited service discipline. To accommodate a more realistic G/G/1/K polling system, we need to formulate an optimization problem first, then we can choose the most flexible and beneficial k-limited service discipline as the decision variable to optimize, say, the mean waiting cost and the mean blocking cost for the polling system. Such an approach is rarely used, because the formulated optimization problem will be a *stochastic simulation optimization* problem with huge discrete solution space, which is computationally intractable and force most researchers to focus on obtaining queuing theory based analytical results with some restrictive assumptions. In other words, such an approach will make sense only if we can solve the formulated problem within a very short period that the arrival rates do not deteriorate visibly. Therefore, the *purpose* of this paper is to solve the optimization problem of a G/G/1/K polling system with k-limited service discipline using limited computation time. To achieve our goal, we will propose an Ordinal Optimization (OO) theory

based approach to find a good enough k-limited service discipline for the G/G/1/K polling system. A special case of the proposed method was presented in [10]. Furthermore, we will provide a *performance analysis* based on the derived models for the proposed method to quantify the *global goodness* of the obtained good enough solution, which is not given in [10] and is one of the major *contributions* of this paper.

We organize our paper in the following manner. In Sect. 2, we will present the formulation for minimizing the mean waiting cost and the mean blocking cost of a G/G/1/K polling system. In Sect. 3, we will describe our OO theory based approach for finding a good enough k-limited service discipline. In Sect. 4, we will test our approach on the M/M/1/K, Γ/Γ/1/K, LN/LN/1/K, and W/W/1/K polling systems and compare the results with those obtained by the other service disciplines. In Sect. 5, we will present the performance analysis of our approach to justify the global goodness of the obtained solution. Finally, we will draw a conclusion in Sect. 6.

2 System Model and Problem Formulation

2.1 G/G/1/K Polling Model

The G/G/1/K polling model considered here consists of a single cyclic server and Q finite queues labeled by $A_1, \ldots, A_Q$ as shown in Fig. 1. This model assumes the following: (i) the probability distribution of the customer arrival process is general; (ii) the arrival rates $\lambda_1, \ldots, \lambda_Q$ may vary with time but do not deteriorate visibly within a very short period, say two minutes; (iii) the probability distribution of the service time is general with service rate μ, and the sequence of service times is i.i.d.;

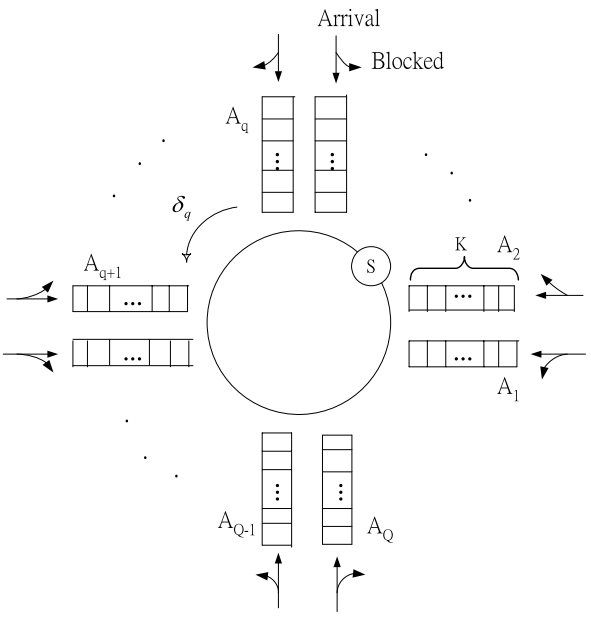

Fig. 1 G/G/1/K polling model

for the sake of simplicity in explanation, we assume μ is constant throughout this paper; (iv) each queue has finite buffer size K.

Remark 2.1 The assumption on the arrival rates described in (ii) is irrelevant to the proposed method but has to do with the execution time, because the arrival rates are assumed constant during the solution process.

We also assume that a customer completing the service will depart from the system, and any customer being served but yet completed will not be interrupted by any reason (non-preemptive). The switchover time incurred when the server switches from A_q to A_{q+1} is assumed to be of *truncated normal* distribution with *parent* mean δ_q, *parent* variance σ_q^2, lower bound 0 and upper bound ∞ and is independent of the arrival processes and the service times. The server continues traveling even when the system is empty.

2.2 Problem Formulation

The service discipline considered in this paper is a k-limited service discipline, which states that the server attends queue A_q, b_q ($\leq K$) customers will be served or until the queue becomes empty, whichever comes first. Therefore the decision variable space of the k-limited service discipline is

$$\Omega = \{b = (b_1, b_2, \ldots, b_Q)^T | 1 \leq b_q \leq K, \text{ and } b_q \text{ is a positive integer for every } q = 1, \ldots, Q\}. \quad (1)$$

We let $\lambda = [\lambda_1, \ldots, \lambda_Q]^T$ denote the vector of arrival rates. We define the *waiting time* of a customer as the time length from arrival instant until the beginning of service, and denote the *random variable* $W_q(b, \lambda)$ as the waiting time of a customer in A_q, which may vary with respect to b and λ. Then, $E[W_q(b, \lambda)]$ represents the mean waiting time of a customer in A_q for a given b and λ. We let τ_q denote the waiting cost parameter of A_q (the cost imposed by having a customer wait for one time unit), then $\frac{1}{Q}\sum_{q=1}^{Q} \tau_q E[W_q(b, \lambda)]$ denotes the mean waiting cost of a customer for the G/G/1/K polling system. As shown in Fig. 1, when a customer arrives while the queue is full, we say that this customer is blocked (or lost). We denote the *random variable* $P_q(b, \lambda)$ as the *blocking rate* or the percentage of customers that are blocked in A_q, which also vary with respect to b and λ. We denote η_q as the blocking cost parameter of A_q (the cost imposed by the blocking rate, i.e. the blocking probability of a customer), then $\frac{1}{Q}\sum_{q=1}^{Q} \eta_q E[P_q(b, \lambda)]$ denotes the mean blocking cost of a customer for the G/G/1/K polling system. Therefore, the expected cost per customer of the system can be stated as $\frac{1}{Q}\sum_{q=1}^{Q}(\tau_q E[W_q(b, \lambda)] + \eta_q E[P_q(b, \lambda)])$, which will serve as the objective function of our optimization problem. In [10], only the mean waiting cost is considered.

Now, we can formulate the optimization problem of the G/G/1/K polling system to find the best k-limited service discipline for a given λ as

$$\min_{b\in\Omega} \frac{1}{Q}\sum_{q=1}^{Q}(\tau_q E[W_q(b,\lambda)] + \eta_q E[P_q(b,\lambda)]). \tag{2}$$

Apparently the optimization problem (2) is a stochastic simulation optimization problem with huge discrete decision variable space Ω shown in (1); Ω is huge because if $K = 20$, then $|\Omega| = K^Q$ or 20^{10} provided that $Q = 10$. However, to evaluate the *true* objective value of a decision vector $b \in \Omega$ for a given λ, we need to perform a stochastic simulation of infinite length, i.e. infinite number of customers for all queues. In fact, this is practically impossible. Therefore, we reformulate (2) as follows:

$$\min_{b\in\Omega} J(b,\lambda)\left(= \frac{1}{Q}\sum_{q=1}^{Q}\left[\tau_q\left(\sum_{m=1}^{L_q} w_q^m(b,\lambda)/L_q\right) + \eta_q(p_q(b,\lambda)/L_q)\right]\right), \tag{3}$$

where L_q denotes the number of arrival customers, $w_q^m(b,\lambda)$ denotes the waiting time of the mth customer, and $p_q(b,\lambda)$ denotes the number of blocked customers at queue A_q, for $q = 1,\ldots,Q$. We let $L = \sum_{q=1}^{Q} L_q$ denote the *simulation length* of (3). Thus, sufficiently large L will make the objective value of (3), $J(b,\lambda)$, sufficiently stable. We let $L_s = Q \times 10000$ represent the sufficiently large L, such that each queue has 10000 customers on the average. In the sequel, we define the *exact model* of (3) as when the simulation length $L = L_s$. For the sake of simplicity in expression, we let $J(b,\lambda)$ denote the objective value of a decision vector b for a given λ computed by exact model, i.e. $L = L_s$.

Remark 2.2 For all the polling models and the randomly generated arrival rates employed in Sect. 4, we found that the values of $J(b,\lambda)$ for $L > Q \times 9800$ vary within 3% range.

2.3 Problem Difficulty

Various heuristic techniques had been developed for solving the simulation optimization problems [11–14]. Among them, genetic algorithm [GA, 15], simulated annealing [SA, 16], and the Tabu search [TS, 17] are frequently used in simulation optimization [18]. Despite the success of several applications of the above heuristic methods [19], many technical hurdles and barriers to broader applications remain as indicated in [20]. Chief among them is speed, because using a lengthy simulation to evaluate the objective value of a discrete decision vector is computationally expensive; also expensive is the search of the best solution, since the space of the discrete decision variables is huge.

3 Solution Method Based on Ordinal Optimization (OO)

3.1 Two-Stage Approach

To cope with the computational complexity of (3), we will employ the OO theory based goal softening strategy [21–23] to efficiently seek a *good enough* k-limited service discipline with *high probability* instead of searching the *best for sure* to make the obtained solution real-time applicable based on the arrival rate assumption given in (ii) of Sect. 2.1.

Remark 3.1 No matter how fast the proposed algorithm could be, it is certainly not instantaneous. However, as long as the execution time is short enough, say two minutes, as posed in the assumption (ii) of Sect. 2.1, we can incorporate with the *time series* forecasting strategy [24] to predict the arrival rate of two minutes later, which will serve as the λ in (3). Due to the limitation on page length, we will not discuss the details of the forecasting strategy in this paper.

Based on the observation that the performance order of the decision vectors is likely preserved even evaluated by a surrogate model; the basic idea of the OO theory is using a computationally easy surrogate-model to quickly evaluate the estimated performance of a decision vector so as to select an *estimated good enough subset* from the *candidate solution set* using limited computation time. If the size of the decision variable space is huge, the reduction of the search space can be done in more than one stage. Our OO theory based approach consists of two stages to solve (3) for a good enough decision vector as outlined below.

The first stage is an *exploration* stage. In this stage, we will employ a typical Genetic Algorithm (GA) to search through Ω using an *off-line* trained Artificial Neural Network (ANN) as a surrogate model for fitness evaluation and select N $(= 1024)$ *roughly* good decision vectors. The second stage is an *exploitation* stage to find a good enough solution from the N solutions obtained in Stage 1 with more refined surrogate models. Suppose we use the *exact model* as defined at the end of Sect. 2.2 to evaluate all the N solutions, we can obtain the best solution in the N, however at the cost of too much computation time, which is against our objective. Therefore, we will divide the second stage into several substages. The surrogate models for *estimating* $J(b, \lambda)$ of a decision vector b for the given λ employed in these substages are stochastic simulations of various lengths ranging from very short (rough model) to very long (exact model). The *candidate* solution set in each substage (or the *estimated* good enough subset resulted from previous substage) will be reduced gradually. In the last substage, we will use the exact model to evaluate all the decision vectors in the most updated candidate solution set, and the one with smallest $J(b, \lambda)$ is the good enough decision vector that we seek. Therefore, the computational complexity can be drastically decreased, because the size of the candidate solution set had been largely reduced when the surrogate model is more refined. In the following, we will present the details of the OO theory based two-stage approach.

3.2 Stage 1: Finding N Roughly Good Solutions from the Huge Discrete Decision Variable Space Ω

Artificial Neural Network (ANN) Model. ANN is considered to be a universal function approximator [25] including the relationship between the input and output of the discrete event simulated systems as presented in [26] and [27]. Thus, treating the decision vector $b \in \Omega$ and the vector of arrival rates λ as input, we wish the output of ANN could estimate $J(b, \lambda)$. To construct such an ANN *off-line*, first of all, we will select a representative subset of Ω by randomly picking B, say 500, decision vectors from Ω and randomly choose Λ, say 1000, vectors of arrival rates λ's with each λ_q ranging from 0.1 to 5 customers/second. Then for each decision vector (b, λ), we will compute the corresponding output $J(b, \lambda)$ using the exact model. These collected $B \times \Lambda$ input-output pairs of $((b, \lambda), J(b, \lambda))$ will be used to train the ANN to adjust its arc weights. The accuracy of the ANN output is closely related to the complicacy of its structure. In other words, there will be a tradeoff between the accuracy and computation time for obtaining the ANN output *on-line*. Since what we care here are the *relative* performance order of b's for a given λ rather than the exact objective value, and we need to use our result *on-line*, we can employ a simple three-layer feedforward ANN structured with $2Q, 4Q$ and 1 neurons in the input, hidden and output layers, respectively. The activation functions of the neurons we employed in the hidden and output layers are the *hyperbolic tangent sigmoid* and *linear* functions, respectively, so as to save computation time. To speed up the convergence of the back propagation training, we used the scaled conjugate gradient algorithm [28] to train the ANN. Stopping criteria of the above training algorithms are when any of the following two conditions occurs: (i) the *sum of the mean squared errors* of the training problem is smaller than 10^{-5}, or (ii) the number of *epochs* exceeds 500. Once this ANN is trained, we can input any vector (b, λ) to obtain an estimation of $J(b, \lambda)$ from the output of the ANN; in this manner, we can avoid using a lengthy stochastic simulation to compute $J(b, \lambda)$ for a given (b, λ).

Genetic Algorithm (GA). By the aid of the above ANN model, we can efficiently search N roughly good decision vectors from Ω using heuristic global searching techniques. Since GA improves the population, a pool of chromosomes, from iteration to iteration, it should best fit our needs. The chromosome in GA terminology represents a decision vector b in our problem, and each decision vector b is encoded by a string of 0s and 1s. We start from I, say 4000, randomly selected decision vectors from Ω as our *initial population*. For the current vector of arrival rates λ, the *fitness* of each decision vector b is set to be the reciprocal of the corresponding estimated $J(b, \lambda)$ based on the ANN. The members in the *mating pool* are selected from the population using *roulette wheel selection scheme*, which selects chromosomes into the mating pool with probabilities proportional to their fitness. We set the probability of selecting members in the mating pool to serve as parents for *crossover* to be p_c. We use a *single point crossover scheme* and assume the *mutation probability* to be p_m. We stop the GA when the number of generations exceeds 20. After the applied GA converges, we rank the final I $(= 4000)$ chromosomes based on their fitness values and pick the top N $(= 1024)$, which are the N *roughly good* decision vectors for the current λ that we look for.

Remark 3.2 To overcome the inaccuracy of ANN, Yen *et al.* had proposed in [29] an integrated simulated annealing and ordinal optimization approach, in which the ANN surrogate model is iteratively refined during the solution process. Technically this approach is sound but not suitable for our problem, because our ANN model need be trained off-line for real-time consideration.

Remark 3.3 Generally speaking, the proposed GA associated with ANN fall into the category of Iterative Ordinal Optimization (IOO) method [30], however, with a flexibility in searching a good enough subset (pool of chromosomes) based on its own searching logic.

3.3 Stage 2: Searching for the Good Enough Decision Vector via a Sequence of Substages

In this stage, we will employ more refined surrogate models than the ANN to evaluate the $J(b,\lambda)$ of a decision vector b for the current λ. These refined surrogate models are stochastic simulations with various length L.

We define a basic simulation length $L_0 = \sum_{q=1}^{Q} \lambda_q t_0$, where t_0 denotes the basic time period for simulation; for example, if $\lambda_q = 1$ customer/second for all q, then setting $t_0 = 1$ minute, we have $L_0 = 600$ customers provided that $Q = 10$. We set the simulation length of substage i, denoted by L_i, to be $L_i = \ell L_{i-1}$ (or $L_i = \ell^i L_0$), $i = 1, 2, \ldots$, where the positive integer ℓ (≥ 2) denotes the parameter for controlling the simulation length L_i. We let $N_0 = N$ and set the size of the selected estimated good enough subset in substage i to be $N_i = N_{i-1}/\ell$ (or $N_i = N_0/\ell^i$), $i = 1, 2, \ldots$. We denote n_ℓ as the total number of substages, and n_ℓ is determined by $n_\ell = \arg\{\min_{n_\ell}(L_0\ell^{n_\ell-1} \leq L_s < L_0\ell^{n_\ell}, 1 < N_{n_\ell-1} \leq 10)\}$, where L_s had been defined at the end of Sect. 2.2. The above formula determines n_ℓ to be the minimum of the following: (i) the n_ℓ such that simulation length $L_0\ell^{n_\ell}$ exceeds the length of exact model, L_s, and (ii) the size of the selected estimated good enough subset resulted in substage $n_\ell - 1$ is small enough, i.e. $1 < N_{n_\ell-1} \leq 10$. Once n_ℓ is determined, we set $L_{n_\ell} = L_s$, which imply that in the last substage (i.e. substage n_ℓ), the surrogate model is in fact the exact model of (3), and the decision vector with smallest $J(b,\lambda)$ is the good enough decision vector that we seek. Suppose ℓ is very large such that $L_1 = \ell L_0 > L_s$, then there will be only one substage, and each of the N decision vectors will be evaluated by the exact model, which will consume too much computation time even though the resulted decision vector is exactly the best among the N as we have indicated in Sect. 3.1. Furthermore, if ℓ satisfies $L_1 = \ell L_0 < L_s$ and $L_2 = \ell^2 L_0 > L_s$, then there will be only two substages as that employed in [10]. Nonetheless, it is not easy to quantify the tradeoff between the computation time and the goodness of the obtained good enough decision vector into an analytical formula. In fact, what is the best ℓ is really problem dependent, because some problems may care more on computation time and some others on the goodness of the obtained solution. Therefore, we will show the computation time and the goodness of the obtained good enough solution of our problem for various ℓ in Sect. 4. Then we will quantify the global goodness of the obtained good enough decision vector corresponding to the most favorable ℓ by the proposed performance analysis in Sect. 5.

3.4 Algorithm Based on Ordinal Optimization (OO)

Now, our OO theory based two-stage approach to solve for a good enough decision vector of (3) can be stated in the following algorithmic steps.

Offline Trained ANN: Randomly select B b's from Ω and randomly choose Λ λ's with each λ_q ranging from 0.1 to 5 customers/second. Compute the corresponding $J(b,\lambda)$ for each (b,λ) using simulation length L_s. Train an ANN by adjusting its vector of arc weights ω using the obtained $B \times \Lambda$ input-output pairs, i.e. the $B \times \Lambda$ pairs of $((b,\lambda), J(b,\lambda))$. Let $f(b,\lambda,\omega)$ denote the functional output of the trained ANN.

Step 1: Fix λ at the current vector of arrival rates. Randomly select I b's from Ω as the initial population. Apply a GA with the following setup: simple roulette-wheel selection scheme, $p_c = 0.7$, single-point crossover scheme and $p_m = 0.02$ to these chromosomes by the aid of the efficient fitness-value evaluation model, $1/f(b,\lambda,\omega)$. After the GA evolves for 20 iterations, we rank all the final I chromosomes based on their fitness values and select the best N. (Note: the values of p_c and p_m employed here are very typical.)

Step 2: For $i = 1$ to $n_\ell - 1$, use the stochastic simulation with simulation length $L_i = \ell^i L_0$ to estimate the $J(b,\lambda)$ of the candidate N/ℓ^{i-1} b's under current λ; rank the candidate N/ℓ^{i-1} b's based on their estimated $J(b,\lambda)$ and select the best N/ℓ^i b's as the candidate solution set for substage $i+1$.

Step 3: Use the stochastic simulation with simulation length L_s to compute the $J(b,\lambda)$ of the candidate $N/\ell^{n_\ell - 1}$ b's. The b with the smallest $J(b,\lambda)$ is the good enough b for the k-limited service discipline under current λ.

4 Test Results and Comparisons

In this section, we will test the computational efficiency[1] of our algorithm and compare the obtained-solution quality with the other service disciplines using four various models: M/M/1/K, Γ/Γ/1/K (Γ = Gamma), LN/LN/1/K (LN = lognormal), and W/W/1/K (W = Weibull) [31]. In the meantime, we will also compare the computational time consumption and the quality of the obtained good enough solutions for various ℓ in our algorithm. To do so, we design three tests. The first one is to show the computational efficiency of our algorithm under a typical arrival rate. The second one is to compare the performance achieved by our algorithm with those obtained by other service disciplines under randomly generated arrival rates but with same cost parameters for all queues. The third test is the same as Test 2, however the cost parameters are randomly generated. All the test results shown in this section are simulated in a Pentium IV PC.

The common polling system parameters in the three tests are set as follows: $Q = 10$, $K = 20$, $\mu = 50$, $\delta_q = 1/30$ and $\sigma_q = 0.01$ for $q = 1, \ldots, Q$. The specific

[1] Note that the other four service disciplines: exhaustive, gated, limited and time-limited are analytical and do not consume any computation time.

Table 1 Good enough k-limited service discipline b^g, obtained $J(b^g, \lambda)$, and CPU time consumed of the algorithm with $\ell = 2$, Test 1

Polling model	k-Limited service discipline										$J(b^g, \lambda)$	CPU time (secs)
	b_1^g	b_2^g	b_3^g	b_4^g	b_5^g	b_6^g	b_7^g	b_8^g	b_9^g	b_{10}^g		
M/M/1/K	8	7	8	8	9	7	8	9	9	8	17.07	109.35
Γ/Γ/1/K	10	9	9	10	8	9	8	8	10	9	13.85	113.85
LN/LN/1/K	10	11	10	12	10	11	10	11	12	10	34.12	118.17
W/W/1/K	9	9	10	11	10	9	11	9	11	10	62.57	112.43

parameters for individual polling model in the three tests are described below. For the Γ/Γ/1/K polling model [31], the shape parameters in the interarrival time pdf, α_{A_q} for $q = 1, \ldots, Q$, and the shape parameter in the service time pdf, α_s, are all set to be 2. For the LN/LN/1/K polling model [31], the standard deviation in the interarrival time pdf, δ_{A_q} for $q = 1, \ldots, Q$, and the standard deviation of the service time pdf, δ_s, are all set to be 0.1. For the W/W/1/K polling model [31], the shape parameters in the interarrival time pdf, β_{A_q}, for $q = 1, \ldots, Q$, and the shape parameter in the service time pdf, β_s, are all set to be 0.5.

The parameters in our algorithm for all tests are set as follows: $B = 500$ and $\Lambda = 1000$ in training the ANN off-line to obtain the vector of arc weights ω for the four various models, $I = 4000$ and $N = 1024$ in Step 1, $L_0 = 540$ and $\ell = 2, 3, 4, 5, 6$ and 200 in Step 2, and $L_s = 100,000$ in Step 3.

In Test 1, we set the arrival rates λ_q, $q = 1, \ldots, Q$ for all four polling models to be 1 customer/second and set the cost parameters τ_q and η_q for $q = 1, \ldots, Q$ to be 10. Based on the above setup of parameters, the good enough k-limited service discipline, denoted by $b^g = (b_1^g, \ldots, b_Q^g)$, and the corresponding $J(b^g, \lambda)$ obtained and the CPU times consumed by our algorithm with $\ell = 2$ for the four various model are shown in Table 1. As can be observed, the consumed CPU times for these four models in this test are all within *2 minutes*, which is short enough that we can incorporate with the time series forecasting technique to apply our algorithm in real-time as indicated in Remark 3.1 of Sect. 3.1.

Remark 4.1 The system considered in Test 1 is a perfectly balance system, because the arrival rates λ_q and the cost parameters τ_q and η_q are the same for all queues. Therefore, the optimal k-limited service discipline, b^*, should have $b_1^* = b_2^* = \cdots = b_Q^*$. Of course, the solution b^g we obtained is not optimal, however, it is good enough as can be observed from Table 1 that the values of $b_1^g, \ldots, b_Q^g$ for each test model are close to each other.

Due to page limitation, we omit the presentation of the obtained good enough k-limited service discipline, $J(b^g, \lambda)$, and the consumed CPU time when $\ell = 3, 4, 5, 6$ and 200. However, interested readers may refer [32]. In general, smaller ℓ corresponds to less CPU time consumption. However, there is no guarantee that larger ℓ will lead to smaller $J(b^g, \lambda)$. Nonetheless, for sufficiently large ℓ such as $\ell = 200$, the corresponding $J(b^g, \lambda)$ is the least and CPU time consumption is the

Table 2 Comparison of the average objective values of the 20 cases for each polling model and each service discipline, Test 2

Discipline	M/M/1/K		Γ/Γ/1/K		LN/LN/1/K		W/W/1/K	
	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$
Our algorithm with $\ell = 2$	40.55	0%	42.08	0%	89.94	0%	146.87	0%
Exhaustive	52.62	29.78%	60.58	43.98%	124.41	38.32%	204.40	39.17%
Gated	73.68	81.71%	85.79	103.89%	162.38	80.54%	227.71	55.04%
Limited	90.43	123.01%	139.21	230.82%	193.26	114.88%	261.55	78.08%
Time-limited	99.56	145.53%	119.44	183.85%	204.46	127.33%	297.79	102.76%

$\bar{J}$: average objective values of the 20 cases
$*$: $\bar{J}$ obtained by the algorithm with $\ell = 2$

longest among all the tested ℓ's as we expect. Therefore, the choice of ℓ is really problem dependent regarding how fast one intends to obtain the solution or how good one cares about the obtained solution. In the considered problem, we prefer to choose $\ell = 2$ for the sake of real-time application.

In Test 2, for each of the four test models, we randomly generate twenty cases of arrival rates, such that the $\lambda_q, q = 1, \ldots, Q$ in each case are randomly generated using U[0.1, 5], where U[*, Δ] represents a *uniform distribution random variable* ranging from * to Δ. In the meantime, we employ same cost parameters τ_q and η_q for $q = 1, \ldots, Q$, as in Test 1. We apply our algorithm with $\ell = 2, 3, 4, 5, 6$ and 200 to the 20 cases of various arrival rates of the four polling models. We also simulate the polling system with the exhaustive, gated, limited, and time-limited (with the server dwelling time on each queue of exponential distribution with mean 2.88 seconds) disciplines to the four models. The resulted average objective values of the 20 cases denoted by $\bar{J}$ for each model and each discipline including our algorithm with $\ell = 2$ are shown in Table 2. In this table, we also show the percentage of the reduction of $\bar{J}$ achieved by our algorithm with respect to the other service disciplines for each polling model. The results show that the performance achieved by our algorithm with $\ell = 2$ is much more superior.

Regarding the effects of various ℓ in this test, we obtain the same conclusion as in Test 1, that is, for real-time application, $\ell = 2$ is preferred. This conclusion on ℓ also applies to Test 3.

In Test 3, we also generate 20 cases for each test model, such that each $\lambda_q, q = 1, \ldots, Q$ is randomly generated using U[0.1, 5] as in Test 2, and each of the cost parameters τ_q and $\eta_q, q = 1, \ldots, Q$ is also generated using U[10, 200] in contrast to using same cost parameters for all queues in Test 2. Applying our algorithm with $\ell = 2$ as well as the other service disciplines to the new set of parameters, the resulted average objective values of the 20 cases, also denoted by $\bar{J}$, for each model and each discipline are shown in Table 3. Comparing the results shown in Tables 2 and 3, we see that the percentage of the reduction of $\bar{J}$ is larger when the cost parameters are different for various queues, which demonstrates that the k-limited service discipline can, indeed, take priority among queues to improve the system performance.

Table 3 Comparison of the average objective values of the 20 cases for each polling model and each service discipline, Test 3

Discipline	M/M/1/K		Γ/Γ/1/K		LN/LN/1/K		W/W/1/K	
	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$	$\bar{J}$	$\frac{\bar{J}-*}{*}100\%$
Our algorithm with $\ell = 2$	107.34	0%	94.92	0%	211.62	0%	388.79	0%
Exhaustive	146.68	36.65%	146.30	54.13%	311.44	47.17%	576.23	48.21%
Gated	195.69	82.31%	226.40	138.52%	393.42	85.91%	616.97	58.69%
Limited	283.41	164.03%	418.44	340.84%	535.76	153.17%	793.56	104.11%
Time-limited	312.75	191.36%	339.06	257.21%	568.14	168.47%	772.37	98.66%

$\bar{J}$: average objective values of the 20 cases
$*$: $\bar{J}$ obtained by the algorithm with $\ell = 2$

To further demonstrate the merit of OO in the aspects of computational efficiency and the obtained-solution quality, we also use GA associated with the exact model alone to solve (3) for the M/M/1/K case in Test 1. The hardest part of using GA alone to solve (3) is the time-consuming objective value evaluation using exact model for each b, which is what OO intends to circumvent and uses a surrogate model, ANN, instead. Due to the time consuming solution process, we terminate the execution of GA associated with the exact model when it consumes 3 hours of CPU time, which is more than 90 times of the CPU time consumed by our approach, and we find that the objective value of their best so far solution (27.51) is still 61.2% more than the objective value of the good enough solution we obtained.

5 Performance Analysis

Although our algorithm outperforms the other service disciplines as demonstrated in Sect. 4, it should be interesting to know how good the solution we obtained is among the search space Ω. Since the primitive candidate solution set considered in our approach is the original solution space, we are facing a different situation from that in [21]. Therefore, to quantify the global goodness of the obtained solution, we need to develop a new performance analysis technique, however still based on the concept presented in [21]. For real-time application consideration, our algorithm is better with $\ell = 2$ as indicated in Sect. 4, thus in the following analysis we set ℓ to be this value.

The methodology for our performance evaluation is to simulate our algorithm based on the Ordered Performance Curves (OPCs) [21], derived from the considered problem, and the employed surrogate models. To proceed with the proposed performance evaluation, we need to (i) analyze the probability distribution of the modeling errors of the employed surrogate models, (ii) investigate the range of the OPCs of the considered problem, and (iii) simulate our algorithm based on the OPCs and the modeling errors. In the following, we will present the details of the analysis, and the derived quantitative results are based on the M/M/1/K polling model with the same

setup of system parameters and algorithmic parameters as in Test 2 of Sect. 4, which will be mentioned in the corresponding analysis.

5.1 Analysis of the ANN Modeling Errors

To analyze the probability distribution of the ANN modeling errors, we randomly select Γ $(= 200)$ decision vectors b's from Ω and randomly select Φ $(= 100)$ vectors of arrival rates λ's such that each λ_q, $q = 1, \ldots, Q$, is randomly selected using U[0.1, 5].

Let $(b(i), \lambda(l))$ denote the ith selected decision vector and the lth selected vector of arrival rates. We first compute $J(b(i), \lambda(l))$ using the stochastic simulation with simulation length L_s for the M/M/1/K polling model with the same setup of system parameters as in Test 2, that are $Q = 10$, $K = 20$, $\mu = 50$, $\delta_q = 1/30$, $\sigma_q = 0.01$, $\tau_q = 10$ and $\eta_q = 10$ for $q = 1, \ldots, Q$. We rank the $\Gamma \times \Phi$ $(b(i), \lambda(l))$'s based on their corresponding $J(b(i), \lambda(l))$'s, such that the smaller the latter, the higher rank the former. We label the $\Gamma \times \Phi$ ordered $(b(i), \lambda(l))$'s by $x_1, \ldots, x_{\Gamma \times \Phi}$ in the sense that x_1 represents the top ranked $(b(i), \lambda(l))$ and plot the $\Gamma \times \Phi$ points of $(x_k, J(x_k))$ in Fig. 2 marked by the solid line. We input each x_k into the ANN, constructed in Test 2 for the M/M/1/K polling model, and let $O(x_k)$ denote the ANN output. We also plot the $\Gamma \times \Phi$ points of $(x_k, O(x_k))$ in Fig. 2 marked by "•". Then, the ANN modeling errors for each x_k denoted by w_k can be calculated by

$$w_k = O(x_k) - J(x_k). \tag{4}$$

Collecting the w_k's for all $x_k, k = 1, \ldots, \Gamma \times \Phi$, we can use a histogram to represent the ANN modeling errors w_k's as shown in Fig. 3. In this figure, the horizontal axis that represents the modeling error w_k is partitioned into the intervals of equal width, 8.33, and the height of a bar represents the number of x_k whose corresponding w_k's lie in the same interval. From Fig. 3, we see that the shape of the histogram is of normal distribution, which leads us to assert the hypothesis that the ANN modeling

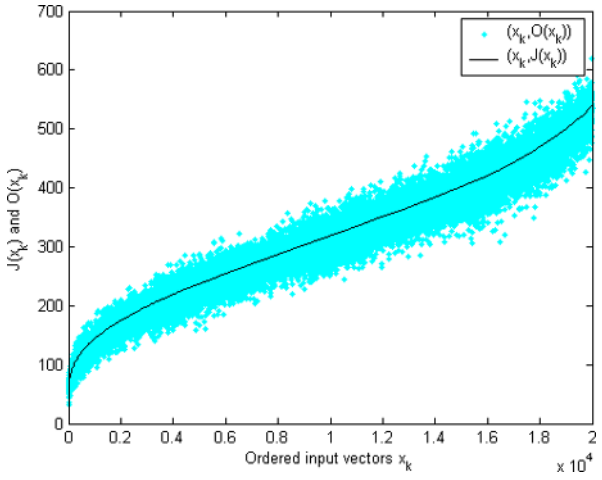

Fig. 2 $J(x_k)$ and $O(x_k)$ for the $\Gamma \times \Phi$ ordered input vectors x_k

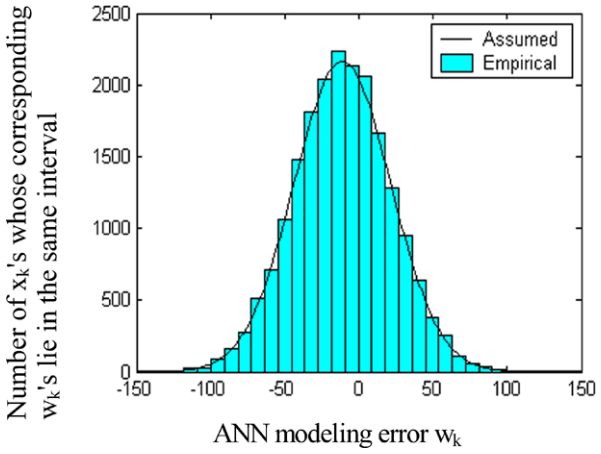

Fig. 3 Histogram of the ANN modeling errors

errors is of a normal distribution with mean μ_w and variance σ_w^2 computed by

$$\mu_w = \sum_{k=1}^{\Gamma\times\Phi} \frac{w_k}{\Gamma \times \Phi}, \tag{5}$$

$$\sigma_w^2 = \sum_{k=1}^{\Gamma\times\Phi} \frac{(w_k - \mu_w)^2}{\Gamma \times \Phi}. \tag{6}$$

How well the model fit the data? Several well-known test methods are available to determine the goodness of fit of a distribution to a set of experimental data, and we will employ the Anderson-Darling test [33] in this paper. We first compute $\mu_w = -11.23$ and $\sigma_w^2 = 1065.16$ based on the $\Gamma \times \Phi$ w_k's obtained by (4). The Anderson-Darling test calculate the Critical Value (CV) and the test statistic AD based on the significance level α, the total number of test data n, the assumed normal distribution F_0 with mean μ_w and variance σ_w^2, and the test data w_k's such that if AD $<$ CV, then the hypothesis is accepted. The formula for calculating CV and AD are given in (7) and (8).

$$CV = C(\alpha)/(1 + 0.75/n + 2.25/n^2), \tag{7}$$

$$AD = -n - \sum_{k=1}^{n} \frac{(2k-1)}{n}\{\ln F_0(w_k) + \ln[1 - F_0(w_{n+1-k})]\}, \tag{8}$$

where $C(\alpha)$ in (7) is a tabulated value of α such that $C(\alpha) = 0.631, 0.752, 0.873$, and 1.035, for $\alpha = 0.1, 0.05, 0.025$, and 0.01, respectively [34]. Setting $\alpha = 0.05$, $n = \Gamma \times \Phi$, we obtain CV $= 0.752$ and AD $= 0.615$, which implies that ANN modeling errors can be modeled by a normal distribution with parameters (μ_w, σ_w^2) computed by (5) and (6) using the data of w_k's obtained from (4).

5.2 Analysis of the Modeling Errors of the Surrogate Models in the Second Stage

The surrogate models of the n_2 (note: $\ell = 2$), which is set to be 8 as we did in Test 2, substages in the second stage are almost the same except for the simulation length.

Therefore, we need only describe the analysis of the modeling errors for one substage, say the first substage. Similar to the analysis process presented in Sect. 5.1, we adopt the $J(x_k)$ of the $\Gamma \times \Phi$ input vectors x_k's shown in Fig. 2 marked by the solid line. We then use the stochastic simulation with simulation length L_1 ($= 2L_0$, where $L_0 = 540$ as we used in Test 2) for the M/M/1/K polling model with the same setup of system parameters as in Test 2, which had been indicated in Sect. 5.1, to estimate $J(x_k)$ for the decision vector x_k. We let $Z(x_k)$ denote the estimated $J(x_k)$ and let e_k denote the modeling error of the surrogate model on x_k, then

$$e_k = Z_k(x_k) - J(x_k). \tag{9}$$

Analyzing the histogram of the modeling error e_k, $k = 1, \ldots, \Gamma \times \Phi$ [32], we also assume the modeling errors of the surrogate model in the first substage to be a normal distribution with mean μ_e and variance σ_e^2 computed by $\mu_e = \sum_{k=1}^{\Gamma\times\Phi} \frac{e_k}{\Gamma\times\Phi} = -7.81$ and $\sigma_e^2 = \sum_{k=1}^{\Gamma\times\Phi} \frac{(e_k-\mu_e)^2}{\Gamma\times\Phi} = 720.66$ based on the data of e_k's obtained from (9). The above probability distribution has passed the Anderson-Darling test at a significance level $\alpha = 0.05$ and sample size $n = \Gamma \times \Phi$ [32]. Thus, the modeling errors of the stochastic simulation with simulation length L_1 is of normal distribution with $\mu_e = -7.81$ (note: $|\mu_e| < |\mu_w|$) and $\sigma_e^2 = 720.66$ ($< \sigma_w^2$), which is indeed more refined than the ANN model in the sense of smaller bias error and smaller variance. A similar process can be carried out for substages $2, 3, \ldots, n_2 - 1$, and the parameters (μ_e, σ_e^2) for the corresponding normal distribution are $(-6.09, 570.66)$, $(-4.47, 397.28)$, $(-3.34, 216.46)$, $(-2.47, 128.65)$, $(-1.63, 69.79)$ and $(-0.92, 27.32)$, respectively. We see that the mean and variance for the n_2 substages become smaller and smaller indicates that the surrogate models are refined substage by substage. Note that we use the exact model (i.e. simulation length equals L_s) in substage n_2.

Remark 5.1 The effects of the modeling errors of any individual stage or substage have been completely reflected in the selected subset of candidate solutions. Thus, the modeling errors of next stage or substage will apply to the previously selected subset of candidate solutions and have nothing to do with the modeling errors of previous stage or substage.

Remark 5.2 Instead of dividing the second stage into several substages, another well-known selection rule in OO is the Optimal Computing Budget Allocation (OCBA) [35], which automatically allocates the next unit of simulation budget among the solution candidates. Under this selection rule, each solution candidate will experience different simulation length. Since the performance analysis is done afterwards, i.e. the simulation length of each solution candidate is already known before performance analysis, thus OCBA can be a good alternative for the present Stage 2 approach, and we can also analyze the corresponding performance.

5.3 Range of the Ordered Performance Curves (OPCs) of (3)

To evaluate the performance of all decision vectors in the decision variable space Ω, we need to consider all possible conditions of the system, and the best way to describe the performance of the decision vectors under a given system condition such

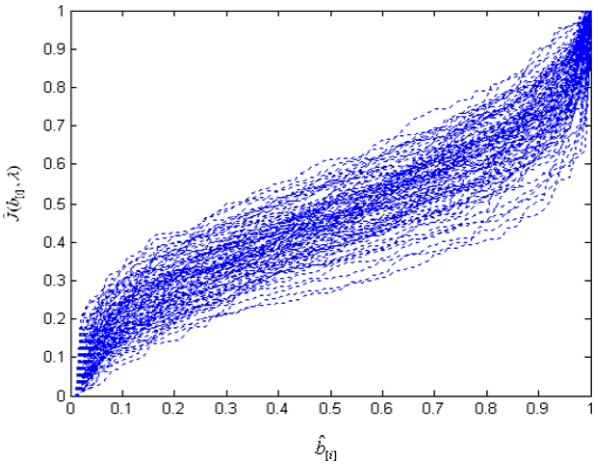

Fig. 4 The range of OPCs of (3)

as the arrival rates λ is using the ordered performance curve (OPC) [21]. To depict the OPC range of the considered problem (3) under various system conditions, we can proceed as follows. We adopt the randomly selected Γ decision vectors b's, the randomly generated Φ vectors of arrival rates λ's, and the corresponding $J(b,\lambda)$'s in the analysis of modeling errors. We let $J_{\max}$ and $J_{\min}$ denote the maximum and minimum $J(b,\lambda)$ among the $\Gamma\times\Phi$ $J(b,\lambda)$'s, respectively, and denote $\hat{J}(b,\lambda)$ as the *normalized* $J(b,\lambda)$ such that $\hat{J}(b,\lambda)=(J(b,\lambda)-J_{\min})/(J_{\max}-J_{\min})$. The values of $\hat{J}(b,\lambda)$'s range from 0 to 1. We order the Γ b's for a given λ according to their $\hat{J}(b,\lambda)$ as $b_{[1]},\ldots,b_{[\Gamma]}$ such that $\hat{J}(b_{[1]},\lambda)\le\hat{J}(b_{[2]},\lambda)\le\cdots\le\hat{J}(b_{[\Gamma]},\lambda)$. We also normalize these ordered Γ $b_{[i]}$'s as $\hat{b}_{[i]}=i/\Gamma$, $i=1,\ldots,\Gamma$ and plot the Γ pairs of $(\hat{b}_{[i]},\hat{J}(b_{[i]},\lambda))$, $i=1,\ldots,\Gamma$ for a given λ to form an OPC. Thus, we can obtain Φ OPCs as shown in Fig. 4. Each of the OPCs can be approximated by a two-parameter, α and β, smooth curve $B^{-1}(z|\alpha,\beta)=B(z|\frac{1}{\alpha},\frac{1}{\beta})$, where $B(z|\cdot,\cdot)$ denotes the Incomplete Beta function of $(\cdot,\cdot)$.

5.4 Performance Evaluation of the Algorithm Based on OO Theory

Now we can simulate our algorithm based on the analyzed modeling errors of the surrogate models employed in Stages 1 and 2 and the possible OPC range of the considered problem.

We let $\hat{\Omega}=\{\frac{1}{|\Omega|},\frac{2}{|\Omega|},\ldots,1\}$ denote the *simulated ordered decision variable space* of Ω, where $|\Omega|$ denotes the cardinality of Ω, such that the kth-order decision vector in $\hat{\Omega}$ is represented by $\frac{k}{|\Omega|}$. Note that we do *not* really evaluate the performance of all decision vectors in Ω. For notational simplicity, we use $\hat{k}$ to denote $\frac{k}{|\Omega|}$, i.e. we set $\hat{k}=\frac{k}{|\Omega|}$.

Our simulation procedures start from picking 20 OPCs which are uniformly distributed within the OPC range described in Fig. 4. We index the 20 OPCs by OPC_n, $n=1,\ldots,20$. Each OPC_n provides a profile of normalized objective values for $\hat{\Omega}$. Starting from OPC_n with $n=1$, we can approximate OPC_n by a $B^{-1}(z|\alpha,\beta)$ with

suitable α and β, say α_n and β_n. We let $\hat{J}(\hat{k})$ denote the *simulated normalized objective value* of $\hat{k}$ for the given OPC_n, then $\hat{J}(\hat{k}) = B^{-1}(\frac{k}{|\Omega|}|\alpha_n, \beta_n)$. To simulate the off-line trained ANN and Step 1 of our algorithm for the given OPC_n, we will *simulate* the *normalized* output of the ANN first. This can be done as follows. For each $\hat{k}$ in $\hat{\Omega}$, we perform the following three steps: (i) compute $\hat{J}(\hat{k}) = B^{-1}(\frac{k}{|\Omega|}|\alpha_n, \beta_n)$, (ii) randomly generate the ANN modeling error w from the model derived in Sect. 5.1 and denote $\hat{w}$ as the *normalized* error such that $\hat{w} = w/J_{\max}$, and (iii) add $\hat{w}$ to $\hat{J}(\hat{k})$. Thus $\hat{w} + \hat{J}(\hat{k})$ represents the *simulated normalized* ANN output and $1/(\hat{w} + \hat{J}(\hat{k}))$ represents the *simulated fitness* of $\hat{k}$. Note that all the $|\Omega|$ points in $\hat{\Omega}$ can be represented by strings of 0's and 1's. Thus to *simulate* Step 1 for $\hat{\Omega}$ with OPC_n, we can apply the same GA as in Step 1 of our algorithm with $I = 4000$, $N = 1024$ and the above mentioned simulated fitness to select N roughly good $\hat{k}$'s from $\hat{\Omega}$.

To simulate Step 2, we need to perform similar simulation processes as that for the off-line trained ANN and Step 1 for each substage in Stage 2 as follows. Starting from $i = 1$, for each of the previously selected N_{i-1} (note $N_0 = N$) estimated good enough $\hat{k}$'s, whose $\hat{J}(\hat{k})$'s are already computed in Step 1, we first randomly generate the modeling error e based on substage i's model derived in Sect. 5.2. We denote $\hat{e}$ as the *normalized* modeling error, such that $\hat{e} = e/J_{\max}$, and add $\hat{e}$ to $\hat{J}(\hat{k})$. The resulted $\hat{e} + \hat{J}(\hat{k})$ is the *simulated* normalized objective value of $\hat{k}$ based on the model employed in substage i. Once $\hat{e} + \hat{J}(\hat{k})$ for all $\hat{k} \in N_{i-1}$ are obtained, we can pick the best $N_i = (N_{i-1}/2)$ $\hat{k}$'s. Repeating the above process for substages $i = 2, 3, \ldots, n_2 - 1$, then we will obtain the candidate solution set N_7.

Now in the last substage, Step 3 of our algorithm, we employ the exact model, i.e. there is no modeling error. Therefore, to simulate Step 3, we can pick the best $\hat{k}$, denote by $\hat{k}_g$, from all $\hat{k} \in N_7$ based on the corresponding $\hat{J}(\hat{k})$'s, which are already computed in the simulated Step 1. $\hat{k}_g$ is the *simulated* good enough decision vector of our algorithm. The order of $\hat{k}_g$ in $\hat{\Omega}$ can be easily told from its value, because $\hat{k} = \frac{k}{|\Omega|}$ that is the k-th ordered decision variable in $\hat{\Omega}$.

Treating the above simulation process as one realization of modeling errors for OPC_n, we simulate 10000 realizations of modeling errors for OPC_n and record the obtained $\hat{k}_g$ for each realization. We order the obtained 10000 $\hat{k}_g$'s as $\hat{k}_g^{[1]}, \hat{k}_g^{[2]}, \ldots, \hat{k}_g^{[10000]}$ in the sense that $\frac{k_g^{[1]}}{|\Omega|} \le \frac{k_g^{[2]}}{|\Omega|} \le \cdots \le \frac{k_g^{[10000]}}{|\Omega|}$, where $\frac{k_g^{[j]}}{|\Omega|}$ is the value of $\hat{k}_g^{[j]}$. Setting $k_g(n) = \hat{k}_g^{[9900]}$, i.e. $k_g(n) = \frac{k_g^{[9900]}}{|\Omega|}$, we conclude that the obtained simulated good enough solution $\hat{k}_g$ is among the best $k_g(n) \times 100\%$ in $\hat{\Omega}$ with probability 0.99 for the given OPC_n.

We repeat the above process for $n = 2, \ldots, 20$, and record $k_g(2), k_g(3), \ldots, k_g(20)$. We found that

$$\bar{k}_g = \frac{1}{20}\sum_{n=1}^{20} k_g(n) \times 100\% = 3.31 \times 10^{-6}\%, \tag{10}$$

$$\sqrt{\frac{1}{20}\sum_{n=1}^{20}\left(k_g(n) - \bar{k}_g\right)^2} \times 100\% = 4.8 \times 10^{-6}\%, \tag{11}$$

which implies that the simulated good enough solution obtained by our algorithm is among the best $3.31 \times 10^{-6}\%$ in $\hat{\Omega}$ with probability 0.99 on the average and with standard deviation $4.8 \times 10^{-6}\%$. The computation burden required to quantify the global goodness of the obtained solution consists of three parts: (i) building the probability distribution of the modeling errors of all the surrogate models, (ii) finding OPC range and (iii) simulating the proposed method. In total, it takes 95.93 seconds of CPU times.

Remark 5.3 We have to admit that we are not able to investigate the true order of the obtained solution in the real solution space Ω, because it is computationally intractable to evaluate the objective values of all $|\Omega|$ ($= 20^{10}$ in our tests) b's. However, the above quantitative result can be a good approximation, because our performance analysis based on the derived models is trustworthy due to (i) the constructed probability distribution of modeling errors and the OPC range have solid foundation in probability theory [33] and the OO theory [21–23], and (ii) we use exactly the same simulation process as the proposed method to carry out the performance analysis.

6 Conclusions

In contrast to the queuing theory based analysis method imposing many restrictive assumptions, we have developed a computationally efficient OO theory based algorithm to solve the optimization problem of a G/G/1/K polling system for a good enough k-limited service discipline. We have not only demonstrated that our algorithm outperforms the other service disciplines but also provided a performance analysis on the global goodness of the good enough solution we obtained. Furthermore, we can incorporate with the time series forecasting strategy [24] to take care of the dynamically varying arrival rates.

Acknowledgement Remark 5.2 is due to an anonymous reviewer.

References

1. Takagi, H.: Analysis and application of polling model. In: Haring, G., Lindemann, C., Reiser, M. (eds.) Performance Evaluation: Origins and Directions. Lecture Notes in Computer Science, vol. 1769, pp. 423–442. Springer, Berlin (2000)
2. Hirayama, T., Hong, S.J., Krunz, M.: A new approach to analysis of polling systems. Queueing Syst. **48**(1–2), 135–158 (2004)
3. Winands, E.M.M., Adan, I.J.B.F., van Houtum, G.J.: Mean value analysis for polling systems. Queueing Syst. **54**(1), 35–44 (2006)
4. Eliazar, I.: Gated polling systems with levy inflow and inter-dependent switchover times: a dynamical-systems approach. Queueing Syst. **49**(1), 49–72 (2005)
5. Leung, K.K.: Cyclic-service systems with nonpreemptive, time-limited service. IEEE Trans. Commun. **42**(8), 2521–2524 (1994)
6. Vishnevskii, V.M., Semenova, O.V.: Mathematical methods to study the polling systems. Autom. Remote Control **67**(2), 173–220 (2006)
7. Takagi, H.: Analysis of finite capacity polling systems. Adv. Appl. Probab. **23**, 373–387 (1991)
8. Jung, W.Y., Un, C.K.: Analysis of a finite-buffer polling system with exhaustive service based on virtual buffering. IEEE Trans. Commun. **42**(12), 3144–3149 (1994)

9. Borst, S.C., Boxma, O.J., Levy, H.: The use of service limits for efficient operation of multistation single-medium communication systems. IEEE/ACM Trans. Netw. **3**(5), 602–612 (1995)
10. Lin, S.-Y., Horng, S.-C.: Ordinal optimization approach to stochastic simulation optimization problems and applications. In: Proc. 15th IASTED Int. Conf. Appl. Simul. Model. Rhodes, Greece, pp. 274–279 (2006)
11. Andradóttir, S.: Simulation optimization: integrated research and practice. INFORMS J. Comput. **14**(3), 216–219 (2002)
12. Ólafsson, S., Kim, J.: Simulation optimization. In: Proc. 2002 Winter Simul. Conf. San Diego, CA, pp. 79–84 (2002)
13. Gosavi, A.: Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Kluwer Academic, Boston (2003)
14. Spall, J.C.: Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. Hoboken, Wiley-Interscience, New Jersy (2003)
15. Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms, 2nd edn. Hoboken, Wiley, New Jersy (2004)
16. Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multiobjective optimization. J. Oper. Res. Soc. **57**(10), 1143–1160 (2006)
17. Hedar, A.R., Fukushima, M.: Tabu search directed by direct search methods for nonlinear global optimization. Eur. J. Oper. Res. **170**(2), 329–349 (2006)
18. Winker, P., Gilli, M.: Applications of optimization heuristics to estimation and modelling problems. Comput. Stat. Data Anal. **47**(2), 211–223 (2004)
19. Fu, M.C., Glover, F.W., April, J.: Simulation optimization: a review, new developments, and applications. In: Proc. 2005 Winter Simul. Conf., Orlando, FL, pp. 83–95 (2005)
20. Tekin, E., Sabuncuoglu, I.: Simulation optimization: a comprehensive review on theory and applications. IIE Trans. **36**(11), 1067–1081 (2004)
21. Lau, T.W.E., Ho, Y.C.: Universal alignment probability and subset selection for ordinal optimization. J. Optim. Theory Appl. **93**(3), 455–489 (1997)
22. Ho, Y.C.: An explanation of ordinal optimization: Soft computing for hard problems. Inf. Sci. **113**(3–4), 169–192 (1999)
23. Ho, Y.C., Zhao, Q.C., Jia, Q.S.: Ordinal Optimization: Soft Optimization for Hard Problems. Springer, New York (2007)
24. Palit, A.K., Popović, D.: Computational Intelligence in Time Series Forecasting Theory and Engineering Applications. Springer, London (2005)
25. Hornik, K., Stinchcombe, M., White, H.: Multilayer fedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
26. Fonseca, D.J., Navaresse, D.O., Moynihan, G.P.: Simulation metamodeling through artificial neural networks. Eng. Appl. Artif. Intell. **16**(3), 177–183 (2003)
27. Alam, F.M., McNaught, K.R., Ringrose, T.J.: A comparison of experimental designs in the development of a neural network simulation metamodel. Simul. Oper. Res. **12**(7–8), 559–578 (2004)
28. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. **6**(4), 525–533 (1993)
29. Yen, C.H., Wong, D.S.H., Jang, S.S.: Solution of trim-loss problem by an integrated simulated annealing and ordinal optimization approach. J. Intell. Manuf. **15**(5), 701–709 (2004)
30. Deng, M., Ho, Y.C.: Iterative ordinal optimization and its applications. In: Proc. 36th IEEE Conf. Decis. Contr., San Diego, CA, vol. 4, pp. 3562–3567 (1997)
31. Banks, J., Carson, J., Nelson, B.L., Nicol, D.: Discrete Event System Simulation, 4th edn. Prentice-Hall, Upper Saddle River (2005)
32. Lin, S.-Y., Horng, S.-C.: Optimization of G/G/1/K polling systems with k-limited service discipline. Technical Report ECE #01, Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan, March (2008)
33. Yazici, B., Yolacan, S.: A comparison of various tests of normality. J. Stat. Comput. Simul. **77**(2), 175–183 (2007)
34. D'Agostino, R.B., Stephens, M.A.: Goodness-of-Fit Techniques. Dekker, New York (1986). Table 4.7, 123
35. Chen, C.H., Lin, J.W., Cesan, E.Y., Chick, S.E.: Simulation budget allocation for further enhancing the efficiency of ordinal optimization. Discret. Event Dyn. Syst. Theory Appl. **10**(3), 251–270 (2000)