USING INTER-ARRIVAL TIMES FOR SCHEDULING IN NON-OBSERVABLE QUEUES

Mikhail Konovalov
Institute of Informatics Problems
of the FRC CSC RAS, Moscow, Russia,
Email: mkonovalov@ipiran.ru

Rostislav Razumchik
Institute of Informatics Problems
of the FRC CSC RAS, Moscow, Russia,
Peoples' Friendship University of Russia
(RUDN University), Moscow, Russia
Email: rrazumchik@ipiran.ru,
razumchik_rv@pfur.ru

KEYWORDS

Queueing system, customer assignment, partial information, deterministic policies, dispatching, simulation

ABSTRACT

In online dispatching systems, when there is no opportunity to observe the state of the systems' components, one may implement "blind" scheduling policies i.e. those which use incomplete/indirect observations of the system state or do not use any information at all. Here we deal with the well-known problem of scheduling in several non-observable parallel single server queues with single Poisson incoming flow, when the broker (scheduler) does not observe neither the current states of the queues and servers, nor the size of the incoming jobs. The only available information is the job size distribution, server's speeds and job's inter-arrival time distribution. For this problem setting it is known that if the scheduler can memorize the sequence of its previous decisions, then a deterministic policy is much better than the probabilistic policy (with respect to the job's mean waiting and mean sojourn time). But if the broker can memorize its decision, it is also very natural to assume that it can also memorize the time instants at which these decisions are made. In this paper we address the following question: can the deterministic policy be improved if the broker, in addition to decisions made, utilizes also the information about the lengths of time between the decisions? We give numerical evidence that it is indeed possible; the cases presented include three, five and nine parallel $\cdot|M|1$ queues. We describe the simple new policy according to which, the broker's decisions are based on the estimates of the queue sizes. In most of the numerical experiments, the new policy outperformed the deterministic policy The relative gain may reach 10% in the case of the job's mean sojourn time and 50% in the case of the job's mean waiting time.

INTRODUCTION

In this paper we consider the following well-known problem of scheduling in parallel non-observable queues. The system consists of N single server infinite capacity queues, operating in parallel. Each server is labelled with a number from 1 to N without repetitions. The service discipline in each server is FCFS. A single flow of jobs arrives at the system. There is a broker (or scheduler), which has to route (immediately[1]) an incoming job to one of the servers. We adopt the following assumptions:

– the scheduler has only static information about the system: CDF of job's inter-arrival times, CDF of job's size and servers' speeds;

– the scheduler can memorize its routing decisions and time instants at which the decisions were made.

Under assumptions made we seek "blind" scheduling policies, which minimize the following cost functions: job's mean waiting and sojourn times in the system. The peculiar feature of the problem is that the online information about the system's state (like queue sizes, remaining work in the servers etc.) is not available to the broker.

There is an extensive literature on this topic. For general and some recent results one can refer to Hordijk, Anneke and Jeroen (1998); Hordijk, Koole and Loeve (1994); Anselmi and Gaujal (2011, 2010); Anselmi, Gaujal and Nesti (2015); Brun (2016); Altman, Gaujal and Hordijk (2000); Gaujal, Hyon and Jean-Marie (2006); Hordijk and van der Laan (2004); Combe and Boxma (1994). Within the considered framework usually two policies attract the most attention: probabilistic and deterministic policies.

According to the probabilistic policy the next job is routed to server m, $1 \leq m \leq N$ with the probability p_m (independently of the previous decisions). For finding p_m there exist efficient numerical procedures (see, for example, Combe and Boxma (1994); Bell and Stidham (1983); Neely and Modiano (2005); Sethuraman and Squillante (1999)).

Deterministic policies, which can achieve significantly better performance than probabilistic policies[2], can be seen as an infinite sequences of numbers $\{a_1, a_2, \ldots, a_{n-1}, a_n, \ldots\}$, where a_m means that m-th job is routed to server a_m. The interest in such policies is justified by the intuitive idea that under such policies the incoming flow to each server is becoming more regular

[1] I.e. the broker does not have a queue to store the jobs.

[2] As, for example, demonstrated in Anselmi and Gaujal (2010).

Proceedings 31st European Conference on Modelling and Simulation ©ECMS Zita Zoltay Paprika, Péter Horák, Kata Váradi, Péter Tamás Zwierczyk, Ágnes Vidovics-Dancs, János Péter Rádics (Editors)
ISBN: 978-0-9932440-4-9/ ISBN: 978-0-9932440-5-6 (CD)

(less random). And according to the folk theorem from queueing theory (see Humblet (1982); Hajek (1983)) determinism in the inter-arrival times minimizes the waiting time in the single server queue. For N queues in parallel, finding the optimal deterministic sequence is a difficult problem. Sometimes a so called billiard sequence can be used instead, which can be constructed using several known greedy algorithms (see Hordijk and van der Laan (2004)). Nevertheless, one still has to estimate the parameters of the algorithms, for which no general solutions to the best of our knowledge exist.

In this paper we propose the algorithm to find policies, which are better than deterministic policies (with respect to the two cost functions considered above) and are still size-oblivious. The key observation for the new algorithm is the following. In order to implement any deterministic policy the broker has to be able to memorize its previous decisions. It is very natural to assume that in addition to memorizing the decisions made, the broker can also memorize the time instants at which the decisions were made. From the technical point of view this cannot be seen as a problem because the broker may be granted access to the internal clock of the system it is implemented in. Thus each decision, say y_n, where to route the n-th customer is associated with the time instant t_n at which it is made (clearly t_n is the customer's arrival epoch). Now when the next job, say $(n+1)$, arrives to the broker it can base its routing decision on the sequence $(y_1, t_1), \dots, (y_n, t_n)$. Yet it does not observe the system's state and even the quantity, which has to be optimized (job's mean waiting or mean sojourn time).

The paper is organized as follows. In the next section we give the description of the exact model considered in this paper. The section is followed by the description of the proposed algorithm for construction the dispatching policy, based on a sequence $(y_1, t_1), \dots, (y_n, t_n)$. Then we present the numerical results provided by the algorithm, and compare them with the results, achieved by known deterministic policies.

MODEL DESCRIPTION

The system consists of N single server infinite capacity queues, operating in parallel. The speed of server m is denoted by v^m, $1 \le m \le N$. The service discipline of each server is FCFS, pre-emption is not allowed. Jobs arrive at the system one-by-one according to a Poisson flow with rate λ. The sizes of the jobs are i.i.d., having exponential distribution with mean 1. Upon arrival each job must be instantly routed to one of the servers. No jockeying between servers and queues is allowed.

DESCRIPTION OF THE ALGORITHM

The key to obtaining the feasible algorithm is the discrete-time setting as in De Vuyst, Bruneel and Fiems (2014). All time-related quantities in the model are discretized into fixed-length intervals of length Δ. Some comments on the choice of Δ will be given in the numerical section and for now we assume that the value of Δ is fixed.

As the job size S has exponential distribution with mean equal to 1, then the service time $S^{(m)}$ in server m has also exponential distribution with rate $v^{(m)}$. We used the following relation to discretize the continuous distribution of $S^{(m)}$:

$$F_0^{(m)} = \mathbf{P}\left\{S^{(m)} = 0\right\} = \mathbf{P}\left\{S < 0,5\Delta v^{(m)}\right\} = 1 - e^{0,5\Delta v^{(m)}},$$

$$F_j^{(m)} = \mathbf{P}\left\{S^{(m)} = j\right\} =$$

$$= \mathbf{P}\left\{S < (m+0,5)\Delta v^{(m)}\right\} - \mathbf{P}\left\{S < (m-0,5)\Delta v^{(m)}\right\} =$$

$$= e^{(m-0,5)\Delta v^{(m)}} - e^{(m+0,5)\Delta v^{(m)}}, \ 1 \le m \le J_m,$$

where $J_m = \min\{J : \sum_{j=0}^{J} F_j^{(m)} > 0,95\}$. The rest of the mass is concentrated in the point $J_m + 1$ and eventually the service times of jobs in server m are i.i.d. with the probability mass function $\{F_j^{(m)}\}$.

Let $0 \le t_1 < t_2 < \dots$ denote the arrival instants of consecutive jobs. Let $y_1, y_2, \dots$ denote the server numbers whereto the jobs are routed i.e. y_n is the server whereto the job arrived at instant t_n was routed. Clearly each y_n takes the value in the set $\{1, 2, \dots, N\}$.

Denote by $u_n^{(m)}$ the workload in the server m at instant $(t_n - 0)$ i.e. just before the job, arrived at instant t_n, was routed to server y_n. By $\tilde{u}_n^{(m)}$ denote the workload in the server m but after the job arrived at instant t_n was routed to server y_n. According to the Lindley equation in queueing theory we have for $n \ge 1$:

$$\tilde{u}_n^{(m)} = u_n^{(m)} + \frac{S}{v^{(m)}} \delta_{m,y_n},$$

$$u_{n+1}^{(m)} = \left(\tilde{u}_n^{(m)} - (t_{n+1} - t_n)\right)^+,$$

where $(\cdot)^+$ is the shorthand notation for $\max(0, \cdot)$ and δ_{ij} is the Kronecker symbol. In this paper we do not pursue any generalizations and assume that initially the system is empty and thus upon arrival of the first job all the servers are empty i.e.

$$u_1^{(1)} = u_1^{(2)} = \dots = u_1^{(N)} = 0. \tag{1}$$

Due to the discrete-time setting the continuous distributions of the workloads $u_n^{(m)}$ and $\tilde{u}_n^{(m)}$ have to be replaced by their discretized versions, which are denoted further by $\{G_{n,k}^{(m)}\}$ and $\{\tilde{G}_{n,k}^{(m)}\}$ i.e.

$$G_{n,k}^{(m)} = \mathbf{P}\left\{u_n^{(m)} = k\right\}, \ k = 0, 1, \dots.$$

$$\tilde{G}_{n,k}^{(m)} = \mathbf{P}\left\{\tilde{u}_n^{(m)} = k\right\}, \ k = 0, 1, \dots.$$

From (1) it follows that the distribution $\{G_{1,k}^{(m)}\}$ is concentrated in one point i.e.

$$G_{1,0}^{(m)} = 1, 1 \le m \le N,$$

$$G_{1,k}^{(m)} = 0, \ k \ge 1, 1 \le m \le N.$$

For any given sequence $y_1, y_2, \dots$, one is able to calculate recursively the probability mass functions $\{G_{n,k}^{(m)}\}$ and $\{\tilde{G}_{n,k}^{(m)}\}$ and the expected workloads $U_n^{(m)} = \mathbf{E}u_n^{(m)}$ using, for example, the approach[3] from De Vuyst, Bruneel and Fiems (2014) (see expressions (7) and (11)). In order to reduce the computational complexity, we had to introduce several refinements into the algorithm. Firstly we have truncated the distributions at a given level (in those numerical results, which are presented here, the distributions were truncated at level below which 95% of the mass is concentrated). Secondly the summations of zero terms were avoided. Below we present relations for the recursive computation of the distributions $\{G_{n,k}^{(m)}\}$ and $\{\tilde{G}_{n,k}^{(m)}\}$. The details are omitted and will appear elsewhere.

Define the following sequence of constants

$$\begin{aligned}
K_n^{(m)} &= \sup\left\{k : G_{n,k}^{(m)} > 0\right\},\\
\tilde{K}_n^{(m)} &= \sup\left\{k : \tilde{G}_{n,k}^{(m)} > 0\right\},\\
\tilde{K}_n^{(m)*} &= \sup\left\{K : \sum_{k=0}^{K} \tilde{G}_{n,k}^{(m)} \geq 0,95\right\},\\
L^{(m)} &= \sup\left\{j : F_j^{(m)} > 0\right\}.
\end{aligned}$$

For $\{\tilde{G}_{n,k}^{(m)}\}$ and $\{G_{n+1,k}^{(m)}\}$ we have for $n \geq 1$:

$$\tilde{G}_{n,k}^{(m)} = \begin{cases} G_{n,k}^{(m)}, & m \neq y_n, \\ \sum\limits_{i=\max(0,k-L^{(m)})}^{\min(k,K_n^{(m)})} G_{n,i}^{(m)} F_{k-i}^{(m)}, & 0 \leq k \leq \tilde{K}_n^{(m)*}, \\ & m = y_n, \end{cases} \qquad (2)$$

$$G_{n+1,0}^{(m)} = \sum_{i=0}^{\min(T_n, \tilde{K}_n^{(m)})} \tilde{G}_{n,i}^{(m)}, \qquad (3)$$

$$G_{n+1,k}^{(m)} = \tilde{G}_{n,k+T_n}^{(m)}, \; 1 \leq k \leq \tilde{K}_n^{(m)} - T_n, \qquad (4)$$

where T_n is integer number nearest to $(t_{n+1} - t_n)/\Delta$.

In order to define the rule for producing the sequence $y_1, y_2, \dots$ we have to come back to the system description. Remind that there is no opportunity to observe neither the queues' sizes nor the remaining work in any server. Let the broker decide where to route the incoming job based on the estimates of the expected workloads $U_n^{(1)}, \dots, U_n^{(N)}$ in each server (including its queue) upon n-th job arrival. The decision rule is the following: send the first job to the fastest server; send the n-th job to the server y_n, where

$$y_n = \operatorname{argmin}_m \left\{U_n^{(m)} + h\frac{1}{\nu^{(m)}}, \; 1 \leq m \leq N\right\}, \; n \geq 2. \quad (5)$$

Here $0 \leq h \leq 1$ is a constant. Due to (1), the expected workload $U_1^{(m)} = 0$ for each m. As the values of the expected workloads $U_n^{(m)}$, $n \geq 2$ can be calculated from (2), (3) and (4) as

$$U_n^{(m)} = \sum_{k=0}^{\tilde{K}_n^{(m)*}} k\, G_{n,k}^{(m)}, \qquad (6)$$

we can write out the algorithm for choosing the next server for an arriving job (see Algorithm 1).

Algorithm 1 Algorithm for choosing the server for an arriving job based on inter-arrival times and estimations of the expected workloads

```
Route the job arrived at instant $t_1$ to the fastest server;
for $i \geq 2$ do
    for job arrived at instant $t_i$ do
        calculate (2) for $n = i - 1$;
        calculate (3), (4) for $n = i - 1$;
        calculate $y_i$ using (5), (6);
        route the job to server $y_i$;
    end for
end for
```

The constant $0 \leq h \leq 1$ which appears in (5), in general, depends on the system's parameters and the cost function. As our experiments show, the value of h allows optimization in the interval [0, 1]. Although in case of minimization of the job's mean waiting time (mean sojourn time) the value of h in (5) has to be equal to zero (one), using the intermediate values sometimes led to better results (lower values of the cost function).

NUMERICAL EXPERIMENT

The purpose of this numerical section is to give the raw comparison of the two scheduling policies – the near optimal deterministic policy and the new policy, proposed in this paper – with respect to job's mean waiting and mean sojourn times.

Some results of such comparison in the system with two servers have been reported in Konovalov and Razumchik (2016). Here we will present the results for the three special cases. The first case is the system with 3 servers with speeds 1, 3 and 7 (see Tables 1 and 2), the second case is the system with 5 servers with speeds 1, 2, 3, 4 and (see Tables 3 and 4) and the third case is the system with 9 servers with speeds 0,9, 1, 1,1, 2,9, 3, 3,1, 6,9, 7 and 7,1 (see Table 5).

In all cases it is assumed that the incoming flow of jobs is Poisson and the job size distribution is exponential with mean equal to 1. Thus the service time distributions are in each case exponential with the rates equal to servers' speeds.

As the near optimal deterministic policy we used the billiard sequence, generated by the SG (Special Greedy) algorithm from (Hordijk and van der Laan, 2004, p.184). In order to construct the dispatching sequence this algorithm takes as input the probabilities p_m of sending an arriving job to server m, $1 \leq m \leq N$ and non-negative rational numbers x_m, $1 \leq m \leq N$. We assumed (as in Anselmi

[3]Another approach and the expression for the expected workload can be found in (Abate and Whitt, 1994, Eq.(18)). Yet in the experiments reported we did not utilize it. Other methods for the calculation of the time-dependent workload can be also found in Stadje (1997); Ackroyd (1986).

and Gaujal (2011)) that $x_m = 1$ if m is the fastest server i.e. if $v_m = \max_j v_j$ and $x_m = 0$ otherwise. In the experiments, results of which we present here, we did not have serves with equal speeds and thus the fastest server had always been unique. The values of p_m were taken as the solution of PA1 problem in Combe and Boxma (1994), when the objective was to minimize the mean waiting time[4]. When the mean sojourn time was to be minimized p_m were taken as the solution of the modified version of the PA1 problem[5].

Sometimes we also report the results for two size-based policies: join-the-shortest-queue (JSQ) and myopic. According to the JSQ policy the arriving job is routed to the server with minimum number of jobs; with the myopic policy the broker chooses such server, which minimizes the sojourn time for the arriving job. We used our own implementation of the JSQ and myopic policies, using the simulation framework Konovalov and Razumchik (2014); Konovalov (2014). The ties in the JSQ policy were broken according to the rule: if two or more servers have the same number of jobs (in the queue and in the server), choose the fastest server. It is worth mentioning that the performance of the JSQ policy heavily depends on how one breaks the ties. If the ties are broken, for example, randomly, the results for the JSQ policy will be much poorer than those presented.

From the tables presented below one can see that the new policy usually outperforms the deterministic policy. The advantage of the new policy is mostly noticeable with respect to the mean waiting time: the relative gain from the new policy may reach 50 %.

Let us consider the results for the 3-server case in more detail (see Tables 1 and 2). In the the 4-th and 5-th rows of the Table 1 one can see the values (rounded to the third decimal place) of the mean waiting times obtained by methods from Combe and Boxma (1994) and Hordijk, Koole and Loeve (1994) and reported in Hordijk, Koole and Loeve (1994). The sign x means that values for such values of the system's load ρ have not been reported in Hordijk, Koole and Loeve (1994). These two methods produce periodic policies. As one can notice these policies are better than the policy, produced by the SG algorithm and worse than the new policy. As the new policy is not periodic, we cannot compare it with respect to the length of the period as done in Hordijk, Koole and Loeve (1994).

From the Table 2 it is worth noticing that policies, which do not use any online information, may outperform size-based policies. Under low system's load, for example $\rho = 0,375$, the relative gain of the deterministic policy (constructed by the SG algorithm) with respect to JSQ policy is $\frac{0,267-0,247}{0,258} \times 100\% \approx 3,3$ % and relative gain of the new policy with respect to JSQ is even higher and equal to $\frac{0,267-0,247}{0,267} \times 100$ % $\approx 7,5$ %. According to our numerical experiments this relative gain monotonically increases with the decrease of the system's load.

The other observation about the new policy is that it is hard to obtain any improvement over the SG algorithm, when the system's load is high (see the last rows of the Tables 1, 2, 3 and 4). Our guess here is that under high load all queues are non-empty most of the time and additional information about the inter-arrival times does not pay any role.

The size of the discretization step size requires more careful treatment and justification. We have obtained the results for the new policy using the discretization step size equal to 0,1. In those experiments with the Algorithm 1, which we report here, making step size smaller did not lead to any improvements.

In the 5-server case (see Tables 3 and 4) the tendency is similar: the new policy is always better than the deterministic one. With respect to the mean waiting time the relative gain is more pronounced.

As the number of servers increases, construction of the dispatching sequence using the new algorithm becomes more challenging: its running time increases and thus it requires high performance computing nodes. It also becomes too sensitive to the value of Δ. Finding good values of p_i for the deterministic policy is another challenge. For the demonstration purposes we have considered the case with 9 servers. The results are presented in the Table 5. Here as well the new policy performs better than the deterministic one. Although for this case we have used the same algorithm as for the two previous cases, our experiments show that the approaches to solve the problems with all distinct servers and with several group of similar servers should not be the same.

SUMMARY

The main result of this paper is the evidence that there is a general opportunity to increase the performance of the non-observable system (of parallel single server queues) by constructing policies, that utilize the decision history more efficiently.

In most of the presented cases the policy proposed in the paper, which utilizes the history of the inter-arrival times, outperforms the near-optimal deterministic policy. The relative gain may reach $\approx 10\%$ with respect to job's mean sojourn time and $\approx 50\%$ with respect to job's mean waiting time.

The improvements that we have demonstrated here of course come at price. The deterministic policy (using SG algorithm) can be implemented in the broker at very limited costs, whereas the new policy requires quite some computational efforts. But whenever there is an opportunity to implement the deterministic policy in the broker, there is also an opportunity to implement the new policy and thus the decision, which one to prefer, is a matter of compromise between costs and benefits.

[4]These values of p_m may not be the optimal ones for the deterministic policy.

[5]The expression for the mean waiting time in queue m was replaced by the expression for the mean sojourn time in queue m.

Table 1: Mean waiting times in the three server system ($N = 3$) under different dispatching policies. Service time is exponential. The service rates are 1, 4 and 7. The arrival flow is Poisson with rate λ.

$\rho = \lambda/12$	SG	MEM			JSQ	Myopic	(SG-MEM)/SG
0,25	0,039	0,024	0,032	0,031	0,007	0,017	38,5 %
0,375	0,073	0,057	x	x	0,026	0,034	21,9 %
0,5	0,128	0,112	0,120	0,119	0,066	0,061	12,5 %
0,625	0,222	0,207	x	x	0,141	0,108	6,7 %
0,750	0,420	0,405	0,412	0,414	0,286	0,208	3,6 %
0,917	1,650	1,650	x	x	1,041	0,852	0%

Table 2: Mean sojourn times in the three server system ($N = 3$) under different dispatching policies. Service time is exponential. The service rates are 1, 4 and 7. The arrival flow is Poisson with rate λ.

$\rho = \lambda/12$	SG	MEM	JSQ	Myopic	(SG-MEM)/SG
0,25	0,220	0,204	0,222	0,175	7,3 %
0,375	0,258	0,247	0,267	0,200	4,3 %
0,5	0,323	0,314	0,321	0,235	2,0 %
0,625	0,440	0,422	0,401	0,294	4,0 %
0,750	0,649	0,632	0,546	0,410	2,7 %
0,917	1,894	1,894	1,296	1,082	0 %

Table 3: Mean waiting times in the system with 5 servers ($N = 5$) under SG and MEM dispatching policies. Service time is exponential. The service rates are 1, 2, 3, 4, 5. The arrival flow is Poisson with rate λ.

$\rho = \lambda/15$	SG	MEM	(SG-MEM)/SG
0,20	0,023	0,011	52,2 %
0,30	0,046	0,032	30,4 %
0,40	0,081	0,067	17,3 %
0,50	0,135	0,122	9,6 %
0,60	0,221	0,210	5,0 %
0,73	0,448	0,437	2,4 %
0,87	1,154	1,145	0,9 %

Table 4: Mean sojourn times in the system with 5 servers ($N = 5$) under SG and MEM dispatching policies. Service time is exponential. The service rates are 1, 2, 3, 4, 5. The arrival flow is Poisson with rate λ.

$\rho = \lambda/15$	SG	MEM	(SG-MEM)/SG
0,20	0,279	0,263	5,7 %
0,30	0,317	0,301	5,0 %
0,40	0,364	0,351	3,6 %
0,50	0,432	0,417	3,5 %
0,60	0,530	0,515	2,8 %
0,73	0,766	0,754	1,6 %
0,87	1,480	1,469	0,7 %

Table 5: Mean waiting and sojourn times in the system with 9 servers ($N = 9$) under SG and MEM dispatching policies. Service time is exponential. The service rates are 0,9, 1, 1,1, 2,9, 3, 3,1, 6,9, 7 and 7,1. The arrival flow is Poisson with rate λ.

	$\rho = \lambda/33$	SG	MEM	(SG-MEM)/SG
mean waiting	0,25	0,018	0,010	44,4 %
times	0,375	0,041	0,032	21,9 %
mean sojourn	0,25	0,187	0,180	3,7 %
times	0,375	0,231	0,222	3,9 %

Acknowledgements This work was supported by the Russian Foundation for Basic Research (grant 15-07-03406).

REFERENCES

Ackroyd, M. 1986. Approximate characterisation of nonstationary discrete time G/G/1 systems. Performance Evaluation. Vol. 6. No. 2. Pp. 117–123.

Abate, J., Whitt W. 1994. Transient Behavior of the M/G/1 Workload Process. Operations Research. Vol. 42. No. 4. Pp. 750–764.

Anselmi, J., Gaujal B. 2010. The price of anarchy in parallel queues revisited. ACM Sigmetrics. Pp. 353–354.

Anselmi, J., Gaujal B. 2011. The price of forgetting in parallel and non-observable queues. Performance Evaluation. Vol. 68. Issue 12. Pp. 1291–1311.

Altman, E., Gaujal B., Hordijk A. 2000. Balanced Sequences and Optimal Routing. Journal of American Computing Machinery. Vol. 47. Issue 4. Pp. 752–775.

Anselmi, J., Gaujal B., Nesti T. 2015. Control of parallel non-observable queues: asymptotic equivalence and optimality of periodic policies. Stochastic Systems. Vol. 5. Issue 1. Pp. 120–145.

Bell, C. H., Stidham S. 1983. Individual versus social optimization in the allocation of customers to alternative servers. Management Science. Vol. 29. No. 7. Pp. 831–839.

Brun, O. 2016. Performance of non-cooperative routing over parallel non-observable queues. Probability in the Engineering and Informational Sciences. Vol. 30. Issue 3. Pp. 455–469.

Combe, M. B., Boxma O. J. 1994. Optimization of static traffic allocation policies. Theor. Comput. Sci. Vol. 125. No. 1. Pp. 17–43.

Gaujal, B., Hyon E., Jean-Marie A. 2006. Optimal routing in two parallel queues with exponential service times. Discrete Event Dynamic Systems. Vol. 16. Issue 1. Pp. 71–107.

Hajek, B. 1983. The proof of a folk theorem on queuing delay with applications to routing in networks. Journal of the ACM. Vol. 30. No. 4. Pp. 834–851.

Humblet, P. 1982. Determinism minimizes waiting time in queues. The Laboratory for Information and Decision Systems Technical Report ser. LIDS-P/1207.

Hordijk, A., Anneke L., Jeroen T. 1998. Analysis of a finite-source customer assignment model with no state information. Mathematical Methods of Operations Research. Vol. 47. Issue 2. Pp. 317–336.

Hordijk, A., Koole G. M. and Loeve J. A. 1994. Analysis of a customer assignment model with no state information. Probability in the Engineering and Informational Sciences. Vol. 8. Pp. 419–429..

Hordijk, A., van der Laan D. A. 2004. Periodic routing to parallel queues and billiard sequences. Math. Method. Oper. Res., 2004. Vol. 59. No. 2. Pp. 173–192.

Konovalov, M., Razumchik R. 2014. Simulation Of Task Distribution In Parallel Processing Systems. Proceedings of the 6th International Congress on Ultra Modern Telecommunications and Control Systems. Pp. 657–663.

Konovalov, M. G. 2014. Building a simulation model for solving scheduling problems of computing resources. Systems and Means of Informatics. Vol. 24. No. 4. Pp. 45–62. (in Russian)

Konovalov, M., Razumchik R. 2016. Dispatching to two parallel nonobservable queues using only static information. Informatics and its applications. Vol. 10. No. 4. Pp. 57–67.

Neely, M. J., Modiano E. 2005. Convexity in queues with general inputs. IEEE Transactions on Information Theory. Vol. 51. No. 2. Pp. 706–714.

Sethuraman, J., Squillante M. S. 1999. Optimal stochastic scheduling in multi-class parallel queues. SIGMETRICS. Pp. 93–102.

Stadje, W. 1997. A new approach to the Lindley recursion. Statistics & Probability Letters. Vol. 31. No. 3. Pp. 169-175.

De Vuyst, S., Bruneel H., Fiems D. 2014. Computationally efficient evaluation of appointment schedules in health care. European Journal of Operational Research. Vol. 237. No. 3. Pp. 1142–1154.

AUTHOR BIOGRAPHIES

MIKHAIL KONOVALOV is a Doctor of Sciences in Technics and holds position of the principal scientist at Information Technologies Department at Institute of Informatics Problems of the Federal Research Center ”Computer Science and Control” of the Russian Academy of Sciences. His research activities are focused on adaptive control of random sequences, modelling and simulation of complex systems. His email address is `mkonovalov@ipiran.ru`.

ROSTISLAV RAZUMCHIK received his Ph.D. degree in Physics and Mathematics in 2011. Since then, he has worked as a leading research fellow at Institute of Informatics Problems of the Federal Research Center “Computer Science and Control” of the Russian Academy of Sciences (FRC CSC RAS). Currently he also holds the associate professor position at Peoples’ Friendship University of Russia (RUDN University). His current research activities are focused on queueing theory and its applications for performance evaluation of stochastic systems. His email address is `rrazumchik@ipiran.ru`