Positioning traffic in NoSQL database systems by the use of particle swarm algorithm

Marcin Woźniak
Institute of Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: Marcin.Wozniak@polsl.pl

***Abstract*—In this paper, application of particle swarm algorithm in positioning and optimization of traffic in NoSQL database is discussed. Sample system is modeled with independent 2-order hyper exponential input stream of packets and exponential service time distribution. Optimization is solved using particle swarm algorithm for various scenarios of operation.**

I. Introduction

In modern computer science, artificial intelligence (AI) as well as evolutionary computing (EC) is one of most important fields, widely applied in various tasks. There are many applications of AI in sciences and industry. The power of such approaches lies within the dedicated mechanisms is used to simulate sophisticated phenomenon. On the other hand some techniques of EC were proven very efficient for searching optimal solutions, easy to implement and precise. Let us give same examples. AI applied to create learning sets are discussed in [1], [2]. Some aspects of positioning computing network models by the use of EC are presented in [3], [4] and [5]. Moreover, AI is used for the optimization of industry processes [6], [7], [8]. AI is also applied for systems which require dedicated solutions [9], [10], [11], as well as for agents oriented programming and object oriented refactoring techniques [12], [13], [14], [15]. In NoSQL systems we use dedicated solutions to increase performance. These applications are especially designed for given purposes, please see [16]. We must build applications to serve clients requests, search in database, maintain service and more. All these aspects demand special mechanisms like dedicated sorting and indexing algorithms and queueing systems to administrate incoming requests. Dedicated sorting algorithms help to organize large data sets as fast as possible. Some examples of dedicated sorting algorithms are discussed in [17], [18], [19] and [20]. While in [18] and [20] is presented dedicated version of quick sort, where implemented modifications enabled faster sorting. In [17] is discussed dedicated merge sort, which even more efficient version is presented in [21]. Moreover extended research on these situations are discussed in [22] and [23], where research on efficient methods of indexing and sorting large NoSQL systems are presented. Here will be discussed another important aspect of optimal service in large NoSQL systems - traffic simulation and positioning.

Traffic in the network and therefore efficient service can increase Quality of Service (QoS) [24]. We can simulate the network traffic, where NoSQL database server is serving various clients. Clients send requests and server collects them to proceed actions. After processing knowledge in database (using methods like [21] or [20]) server responses to the requests, but this goes according to the income queue. Earlier requests must be served first and others according to arrival time. The problem is to position this system for most efficient operation (we shall define optimal service, vacation and income parameters). In this paper NoSQL database system will be positioned for most efficient service and lowest possible cost of work by the use of particle swarm optimization algorithm (PSO).

II. Applied mass service model

For NoSQL database systems various methods of modeling and simulation can be applied. Mainly we try to analyze the model, which describes operation. Operation model is defined for applied queueing system (QS). Service description of NoSQL database, where dedicated QS is applied to optimize operation cost defines $T_{service}$, T_{income} and $T_{vacation}$, which describe average time of service, average income time and average vacation time (backup, conservation and etc.), respectively. All these are independent random variables, where the symbols in time t are:

- τ_1 — the first busy period starting at $t = 0$;
- δ_1 — the first idle time (first vacation time and first standby time);
- $h(\tau_1)$ — the number of packets served during τ_1;
- $X(t)$ — the number of packets in the system at t.

In this paper is discussed simulation and positioning of NoSQL database traffic modeled with dedicated QS, where we define only one request arrival and response departure.

A. Analytical results

Modeling traffic in NoSQL systems is non-trivial problem. Classical cost structure is considered in [25]. While in [26], [27], [28] are presented most important aspects of positioning and cost optimization. Various queueing models for applied type of the server are investigated in [29], [30], [31], [32], [33], [34], [35]. Please see also [36] and [37] for a review of important results on modeling and positioning.

In this paper are applied results of the research on similar objects, see [38] and [39] for joint transform of first busy period, first idle time and number of packets completely served during first busy period in $GI/G/1$-type systems. More on generally distributed service times and infinite buffers can be found in [40] and [41]. All these research results are helpful to model and position QS of different type as discussed in [42] [43], [4] and [44]. Where in [42] or [43] was given an idea to apply evolutionary computation (EC) in QS simulation and positioning. An extension of the research for sophisticated QS were published in [4]. And finally main analytical model with detailed description and assessments for traffic in the system was given in [44]. Let us see the model of QS for NoSQL database traffic.

To model NoSQL server operation was used a finite-buffer $H_2/M/1/N$-type QS, similar to server traffic modeling functions discussed in [45] and [46]. Let it be here presented only a brief description, just to help in understanding NoSQL positioning and simulation problem (for details please see [44]). Incoming requests describes 2-order distribution function:

$$F(t) = p_1\big(1 - e^{-\lambda_1 t}\big) + p_2\big(1 - e^{-\lambda_2 t}\big), \quad t > 0, \tag{1}$$

where $\lambda_i > 0$ for $i = 1, 2$ and $p_1, p_2 \geq 0$. Inter-arrival times are mixed of two exponential distributions with parameters λ_1 and λ_2, which are being “chosen” with probabilities p_1 and p_2. In the system, there are $(N - 1)$ places in queue and one for packet in the service. System starts working at $t = 0$ with at least one packet present. After busy period the server begins vacation which is modeled with 2-order hyper exponential distribution function:

$$V(t) = q_1\big(1 - e^{-\alpha_1 t}\big) + q_2\big(1 - e^{-\alpha_2 t}\big), \quad t > 0. \tag{2}$$

Interpretation of parameters α_i, $i = 1, 2$ and q_1, q_2 is similar to that for λ_i, $i = 1, 2$ and p_1 and p_2. If at the end of vacation there is no packet present in the system, the server is on standby and waits for first arrival to start service process. If there is at least one packet waiting for service in the buffer at the end of vacation, the service process starts immediately and new busy period begins.

Functions $F(\cdot)$ and $V(\cdot)$ help to simulate operation of the examined NoSQL system, where inter-arrival times and vacation are defined in (1) and (2). In the research PSO is used to find optimal set of parameters λ_i, p_i, μ and α_i. To describe minimal amount of resources to perform all operations $r_n(c_1)$ is defined:

$$r_n(c_1) = \frac{Q_n(c_1)}{\mathbf{E}_n(c_1)} = \frac{r(\tau_1)\mathbf{E}_n\tau_1 + r(\delta_1)\mathbf{E}_n\delta_1}{\mathbf{E}_n\tau_1 + \mathbf{E}_n\delta_1}, \tag{3}$$

where the symbols are: $r(\tau_1)$-fixed unit operation costs during busy period τ_1, $r(\delta_1)$-fixed unit operation costs during idle time δ_1, $\mathbf{E}_n\tau_1$-means of busy period τ_1 and $\mathbf{E}_n\delta_1$-idle time δ_1 on condition that system starts with n packets present. In (3) are used means of busy period and vacation (idle) time. The explicit formula with detailed information and description for conditional joint characteristic functions of τ_1, δ_1 and $h(\tau_1)$ is presented in [4] and [44]. Here let us briefly discuss modeling of applied QS. General equation to calculate this values is:

$$B_n(s, \varrho, z) = \mathbf{E}\{e^{-s\tau_1 - \varrho\delta_1} z^{h(\tau_1)} \,|\, X(0) = n\}, 2 \leq n \leq N, \tag{4}$$

where $s \geq 0$, $\varrho \geq 0$ and $|z| \leq 1$, $n \geq 1$. Details on this equation are discussed in [17], [4] and [44], where using it we can define, components of (3) total cos of work:

$$\mathbf{E}_n e^{-s\tau_1} = \mathbf{E}\{e^{-s\tau_1} \,|\, X(0) = n\} = B_n(s, 0, 1), \tag{5}$$

then for model of traffic finally we have:

$$\mathbf{E}_n\tau_1 = -\frac{\partial}{\partial s} B_n(s, 0, 1)\Big|_{s=0}, \tag{6}$$

similarly we have:

$$\mathbf{E}_n\delta_1 = -\frac{\partial}{\partial \varrho} B_n(0, \varrho, 1)\Big|_{\varrho=0}. \tag{7}$$

III. Applied Particle Swarm Optimization algorithm

Particle swarm optimization algorithm (PSO) has been shown in [47]. In the initial form PSO was modeling behaviors that can be observed in young birds or fish, which in the cluster behave in a very specific way. Thanks to the ease of implementation and adaptation to different tasks PSO algorithm has become one of the most commonly used algorithms in it’s original or modified versions. In [48] is presented adaptivity of this methods to different initial conditions of positioned object. While in [49], [50] and [51] many possible aspects of application of various EC methods in engineering optimization are discussed. In [52] PSO application in relay times is presented. Finally discussion on theoretical aspects of convergence and stability can be found in [53], [54] and [55]. Let us discuss behavior of typical swarm.

In the swarm similar operations are performed by many individuals of the same species. In action, individuals communicate with each other in a manner characteristic for the species. Communication helps to exchange information and as a result the whole swarm is moving in a certain direction or behaves like one big organism. PSO algorithm uses the insights that emerge from the observation of swarms of fish or insects that are looking for food or a safe place. This process can be described in mathematical model. If we accept the goal of optimizing criterion function of the object, we can talk about optimizing algorithm.

PSO algorithm searches the space of test solutions by matching trajectories of individuals (particles) in a quasi-stochastic way. A particular individual is a vector and it’s movement is the result of stochastic and deterministic components of movement model. Stochastic component corresponds to random walk. In contrast, deterministic component of the movement model is distance between particles, or other feature, which is modeled in mathematical equation. In subsequent periods individual particles move in looking for the global optimum, where because of stochastic component this movement also has a random character. This combination gives ability to efficiently search the test area of the simulated or positioned object. If during motion the particle is on a new position, which is characterized by better properties of the optimum, for this position it updates the knowledge. In further exploration particle accepts found value as the optimum and starts searching in relation to this value. In each iteration of the algorithm, the particles can communicate with each other and share information about the sought optimum. If we consider that all particles in the swarm want to reach the sought optimum criterion function for the positioned object, in the end we can take as the optimum best of all-values. In this way, entire swarm is communicating between it’s individuals while looking for the global optimum of the criterion function.

A. PSO model

Actions taken while searching for the optimum criterion function of the object are written as mathematical equations. The model of particle swarm movement keeps the communication between particles based on a deterministic factor, but also introduces randomness of the movements. To build the model of the swarm behavior in the solution space of the object are used the following assumptions:

- Points in the search space are seen as potential solutions to moving particle swarm.
- Each particle is seeking for optimum, which is determined by it's position in the space.
- At the end of PSO iteration, the particles interact with other particles and change information.
- As a result of communication global optimum is selected, relative to which all particles are continuing their search.
- Number of moving particles is determined.

In the model, we mean a particle moving in a virtual way. We only model the choice of the optimum to which particle has moved. Selected points in the study area are compared, and among them is chosen the global optimum, see also [56] for details on convergence and stability of PSO.

PSO algorithm for each particle takes the form of $\mathbf{x}_i^t$ whose i components correspond to dimensions of the test space. Each particle is moving at the speed $\mathbf{v}_i^t$ appropriate for the swarm in a particular PSO iteration. These values vary in subsequent iterations of the algorithm. Speed of movement of the particles is described by the formula:

$$v_i^{t+1} = v_i^t + \alpha \cdot \epsilon_1 \cdot [g_*^t - f(x_i^t)] + \beta \cdot \epsilon_2 \cdot [x_*^t - x_i^t], \qquad (8)$$

where the symbols are: v_i^t–speed of i particle in t iteration, α–optimum value memory factor, β–optimum position memory factor, $\epsilon_1, \epsilon_2 \in [0,1]$–random values, g_*^t–optimum for t iteration, x_*^t–optimum position for t iteration, $f(x_i^t)$– fitness function value for i particle in t iteration, x_i^t–position of i particle in t iteration.

Equation repositioning particle swarm movement in each iteration of the PSO algorithm is defined using formula:

$$x_i^{t+1} = x_i^t + (-1)^K \cdot v_i^t, \qquad (9)$$

where the symbols are: x_i^t–position of i particle in t iteration, v_i^t–speed of i particle in t iteration, K–random factor to change motion direction. The initial coordinates of the particle swarm position and their speed we take at random. However, it is possible also to apply some boundary criteria that will allow additional control of the swarm.

These two equations allow to change position of each particle and therefore search entire space for the optimum of the modeled object. Let us now see possible implementation of PSO, which is presented in Algorithm 1.

Algorithm 1 Basic PSO applied to position NoSQL database system traffic

```
1:  Define all coefficients: $\alpha$–optimum value memory factor, $\beta$–optimum position memory factor, $generation$– number of iterations in the algorithm, $particles$–number of particles in the swarm,
2:  Dedicated criterion function: lowest cost of NoSQL system operation (3),
3:  Create at random initial population,
4:  t:=0,
5:  while $t \leq generations$ do
6:      Move $particles$ according to (9) and (8),
7:      Sort $particles$ according to the value of criterion function,
8:      Evaluate population and take $best\_ratio$ of them to next $generation$,
9:      Rest of $particles$ take at random,
10:     Next $generation$: $t++$,
11: end while
12: Best $particles$ from the last $generation$ are potential optimum.
```

IV. Research results

Research results help to predict possible response time and optimize service cost $r_n(c_1)$ considered in different variants: under-load, critical load and overload. PSO simulations were performed for $r(\tau_1) = 0.5$ and $r(\delta_1) = 0.5$. It means that modeled NoSQL database system uses 0.5 energy unit each vacation and work period. For other system types these values may be changed in (3), what makes presented model flexible and easily applicable. All presented research results are averaged values of 100 PSO samplings for 20 particles in 80 iterations with $\alpha = 0.4$ and $\beta = 0.4$. In each iteration $best\_ratio = 90\%$, what means that 72 best particles were moved to next generation and 8 were taken at random. This helped to search entire object space for optimum values, where:

- Average service time: $T_{service} = \frac{1}{\mu}$,
- Average time between packages income into the system: $T_{income} = \frac{p_1}{\lambda_1} + \frac{p_2}{\lambda_2}$,
- Average vacation time: $T_{vacation} = \frac{q_1}{\alpha_1} + \frac{q_2}{\alpha_2}$,
- Examined system size: N = buffer size +1.

Scenario 1.
PSO was performed to find set of parameters for lowest cost of work. In Table I are optimum values for all parameters that affect NoSQL server work. PSO positioned NoSQL system to operate at minimum costs, if the service and vacation are results from Table I. PSO was also arranged to position the system in various scenarios.

TABLE I. OPTIMAL PARAMETERS $\mu, \lambda_i, \alpha_i, p_i, q_i$ FOR $i = 1, 2$ AND LOWEST VALUE OF (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
2.9	2.3	1.43	0.32	1.78	1.3	6.1	3.5
μ	0.6			$r_n(c_1)$	0.34		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	1.67		1.18		15.20		

Scenario 2.
NoSQL $T_{service} = 2[sec]$, what means that request service takes about $2[sec]$. Research results are shown in Table II.

TABLE II. OPTIMAL PARAMETERS μ, λ_i, α_i, p_i, q_i FOR $i = 1, 2$ AND LOWEST VALUE OF (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
2.13	3.15	0.94	0.78	79.70	0.89	2.30	12.10
μ	0.5			$r_n(c_1)$	0.37		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	2.08		37.70		17.96		

Scenario 3.
NoSQL $T_{service} = 0.5[sec]$. This situation represents NoSQL business service with heavy traffic and very efficient server machine. Research results with system positioning are shown in Table III.

TABLE III. OPTIMAL PARAMETERS μ, λ_i, α_i, p_i, q_i FOR $i = 1, 2$ AND LOWEST VALUE OF (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
44.3	22.1	112.9	1.4	1.91	1	57.60	14.3
μ	2.00			$r_n(c_1)$	0.29		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	0.5		0.09		10.72		

Using PSO it is also possible to position the system for T_{income}. This will correspond to peculiar incoming traffic.
Scenario 4.
NoSQL T_{income} was given as $2[sec]$, what means that requests are incoming to the server once in every 2 seconds. Research results are shown in Table IV.

TABLE IV. OPTIMAL PARAMETERS μ, λ_i, α_i, p_i, q_i FOR $i = 1, 2$ AND LOWEST VALUE OF (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
3.7	4.5	1.3	1.4	6.2	2.5	13.1	7.2
μ	0.4			$r_n(c_1)$	0.43		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	2.5		2.23		15.22		

Scenario 5.
NoSQL T_{income} was given as $0.5[sec]$, what means that requests are incoming to the server twice in every second. This situation is describing an extensively used database, like these of business purpose. Research results are shown in Table V.

TABLE V. OPTIMAL PARAMETERS μ, λ_i, α_i, p_i, q_i FOR $i = 1, 2$ AND LOWEST VALUE OF (3).

λ_1	λ_2	α_1	α_2	p_1	p_2	q_1	q_2
27.1	27.2	0.8	0.4	16.1	1.1	5.6	4.3
μ	0.7			$r_n(c_1)$	0.33		
	$T_{service}$		T_{income}		$T_{vacation}$		
$[sec]$	1.43		0.63		17.75		

V. CONCLUSIONS

Positioned model was simulated in situations with predefined time of service or time of income. Given times reflect situations when traffic is heavy and system must serve many requests (like business machines) or common traffic (like average on-line shop or customer service). Calculated values of $T_{service}$ and T_{income} gave positioning for lowest cost of work. If the system works with calculated parameters QoS is still very high, but also cost of service is possibly lowest, what means better profit for the owner. In the article, have been examined newly proposed methods for QS simulation and positioning (see also [4] and [44]). EC methods like PSO are excellent for simulation or positioning of different objects. PSO method helps to simulate complicated objects and because of the free design, calculations are easy to perform. The experiments confirmed PSO efficiency in simulation and positioning the system in various scenarios representing common traffic situations.

REFERENCES

[1] G. Capizzi, C. Napoli, and L.釉ternò, "An innovative hybrid neuro-wavelet method for reconstruction of missing data in astronomical photometric surveys," in *Artificial Intelligence and Soft Computing*. Springer Berlin Heidelberg, 2012, pp. 21–29.

[2] C. Napoli, F. Bonanno, and G. Capizzi, "An hybrid neuro-wavelet approach for long-term prediction of solar wind," *IAU Symposium*, no. 274, pp. 247–249, 2010.

[3] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D. Połap, and M. Woźniak, "Simplified firefly algorithm for 2d image key-points search," in *IEEE Symposium Series on Computational Intelligence*. IEEE, 2014.

[4] M. Woźniak, W. M. Kempa, M. Gabryel, R. K. Nowicki, and Z. Shao, "On applying evolutionary computation methods to optimization of vacation cycle costs in finite-buffer queue," *Lecture Notes in Artificial Intelligence - ICAISC'2014*, vol. 8467 (PART I), pp. 480–491, 2014.

[5] C. Napoli, F. Bonanno, and G. Capizzi, "Exploiting solar wind time series correlation with magnetospheric response by using an hybrid neuro-wavelet approach," *Proceedings of the International Astronomical Union*, vol. 6, no. S274, pp. 156–158, 2010.

[6] G. Capizzi, F. Bonanno, and C. Napoli, "Hybrid neural networks architectures for soc and voltage prediction of new generation batteries storage," in *Clean Electrical Power (ICCEP), 2011 International Conference on*. IEEE, 2011, pp. 341–344.

[7] F. Bonanno, G. Capizzi, A. Gagliano, and C. Napoli, "Optimal management of various renewable energy sources by a new forecasting method," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*. IEEE, 2012, pp. 934–940.

[8] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A cascade neural network architecture investigating surface plasmon polaritons propagation for thin metals in openmp," *Lecture Notes in Artificial Intelligence - ICAISC'2014*, vol. 8468, PART I, pp. 22–33, 2014.

[9] G. Capizzi, F. Bonanno, and C. Napoli, "Recurrent neural network-based control strategy for battery energy storage in generation systems with intermittent renewable energy sources," in *Clean Electrical Power (ICCEP), 2011 International Conference on*. IEEE, 2011, pp. 336–340.

[10] F. Bonanno, G. Capizzi, and C. Napoli, "Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*. IEEE, 2012, pp. 941–945.

[11] G. Capizzi, F. Bonanno, and C. Napoli, "A new approach for lead-acid batteries modeling by local cosine," in *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*. IEEE, 2010, pp. 1074–1079.

[12] C. Napoli, G. Pappalardo, and E. Tramontana, "Using modularity metrics to assist move method refactoring of large systems," in *Seventh International Conference on Complex, Intelligent, and Software Intensive Systems - CISIS 2013*, July 2013, pp. 529–534.

[13] R. Giunta, G. Pappalardo, and E. Tramontana, "AODP: refactoring code to provide advanced aspect-oriented modularization of design patterns,"

in *Proceedings of Symposium on Applied Computing (SAC)*. ACM, 2012.

[14] E. Tramontana, "Detecting extra relationships for design patterns roles," in *Proceedings of AsianPlop*, March 2014.

[15] G. Pappalardo and E. Tramontana, "Suggesting extract class refactoring opportunities by measuring strength of method interactions," in *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*. IEEE, December 2013.

[16] Z. Marszałek and M. Woźniak, "On possible organizing nosql database systems," *International Journal of Information Science and Intelligent System*, vol. 2, no. 2, pp. 51–59, 2013.

[17] M. Woźniak, Z. Marszałek, M. Gabryel, and R. K. Nowicki, "Modified merge sort algorithm for large scale data sets," *Lecture Notes in Artificial Intelligence - ICAISC'2013*, vol. 7895 (PART II), pp. 612–622, 2013.

[18] ——, "On quick sort algorithm performance for large data sets," in *Looking into the Future of Creativity and Decision Support Systems*, A. M. J. Skulimowski, Ed. Cracow, Poland: Progress & Business Publishers, 2013, pp. 647–656.

[19] ——, "Triple heap sort algorithm for large data sets," in *Looking into the Future of Creativity and Decision Support Systems*, A. M. J. Skulimowski, Ed. Cracow, Poland: Progress & Business Publishers, 2013, pp. 657–665.

[20] ——, "Preprocessing large data sets by the use of quick sort algorithm," *Advances in Intelligent Systems and Computing - KICSS'2013*, vol. accepted–in press, 2014.

[21] Z. Marszałek, D. Połap, and M. Woźniak, "On preprocessing large data sets by the use of triple merge sort algorithm," in *Proceedings of International Conference on Advances in Information Processing and Communication Technologies - IPCT'2014*. Santa Barbara, California, USA: The IRED, Seek Digital Library, 2014, pp. 65–72.

[22] M. Woźniak and Z. Marszałek, *Selected Algorithms for Sorting Large Data Sets*. Gliwice, Poland: Silesian University of Technology Press, 2013.

[23] ——, *Extended Algorithms for Sorting Large Data Sets*. Gliwice, Poland: Silesian University of Technology Press, 2014.

[24] C. Napoli, G. Pappalardo, and E. Tramontana, "A hybrid neurowavelet predictor for qos control and stability," in *Proceedings of AI*IA*, ser. LNCS, vol. 8249. Springer, 2013, pp. 527–538.

[25] J. Teghem, "Control of the service process in a queueing system," *European Journal of Operations Research*, vol. 1, no. 23, pp. 141–158, 1986.

[26] O. Kella, "Optimal control of the vacation scheme in an m/g/1 queue," *Operations Research Journal*, vol. 4, no. 38, pp. 724–728, 1990.

[27] R. Lillo, "Optimal operating policy for an m/g/1 exhaustive server-vacation model," *Methodology and Computing in Applied Probability*, vol. 2, no. 2, pp. 153–167, 2000.

[28] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana, "A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*. IEEE, 2014, pp. 1077–1084.

[29] U. C. Gupta, A. D. Banik, and S. Pathak, "Complete analysis of map/g/1/n queue with single (multiple) vacation(s) under limited service discipline," *Journal of Applied Mathematics and Stochastic Analysis*, no. 3, pp. 353–373, 2005.

[30] U. C. Gupta and K. Sikdar, "Computing queue length distributions in map/g/1/n queue under single and multiple vacation," *Applied Mathematics and Computation*, vol. 2, no. 174, pp. 1498–1525, 2006.

[31] C. Napoli, G. Papplardo, and E. Tramontana, "Improving files availability for bittorrent using a diffusion model," in *IEEE 23nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises - WETICE 2014*, June 2014, pp. 191–196.

[32] F. Banno, D. Marletta, G. Pappalardo, and E. Tramontana, "Tackling consistency issues for runtime updating distributed systems," in *Proceedings of International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE, 2010, pp. 1–8.

[33] E. Tramontana, "Automatically characterising components with concerns and reducing tangling," in *Proceedings of QUORS workshop at Compsac*. IEEE, 2013.

[34] Z. Niu and Y. Takahashi, "A finite-capacity queue with exhaustive vacation/close-down/setup times and markovian arrival processes," *Queueing Systems*, vol. 1, no. 31, pp. 1–23, 1999.

[35] Z. Niu, T. Shu, and Y. Takahashi, "A vacation queue with setup and close-down times and batch markovian arrival processes," *Performance Evaluation Journal*, vol. 3, no. 54, pp. 225–248, 2003.

[36] H. Takagi, *Queueing Analysis, vol. 1: Vacation and Priority Systems, vol. 2. Finite Systems*. Amsterdam: North-Holland, 1993.

[37] N. Tian and Z. G. Zhang, *Vacation queueing models. Theory and applications*. Berlin, Heidelberg: Springer - Verlag, 2006.

[38] W. M. Kempa, "Gi/g/1/ batch arrival queuing system with a single exponential vacation," *Mathematical Methods of Operations Research*, vol. 1, no. 69, pp. 81–97, 2009.

[39] ——, "Characteristics of vacation cycle in the batch arrival queuing system with single vacations and exhaustive service," *International Journal of Applied Mathematics*, vol. 4, no. 23, pp. 747–758, 2010.

[40] ——, "Some new results for departure process in the m/g/1 queuing system with a single vacation and exhaustive service," *Stochastic Analysis and Applications*, vol. 1, no. 28, pp. 26–43, 2009.

[41] ——, "On departure process in the batch arrival queue with single vacation and setup time," *Annales UMCS Informatica*, vol. 1, no. 10, pp. 93–102, 2010.

[42] M. Gabryel, R. K. Nowicki, M. Woźniak, and W. M. Kempa, "Genetic cost optimization of the gi/m/1/n finite-buffer queue with a single vacation policy," *Lecture Notes in Artificial Intelligence - ICAISC'2013*, vol. 7895 (PART II), pp. 12–23, 2013.

[43] M. Woźniak, "On applying cuckoo search algorithm to positioning gi/m/1/n finite-buffer queue with a single vacation policy," in *Proceedings of the 12th Mexican International Conference on Artificial Intelligence - MICAI'2013*. IEEE, 2013, pp. 59–64.

[44] M. Woźniak, W. M. Kempa, M. Gabryel, and R. K. Nowicki, "A finite-buffer queue with single vacation policy - analytical study with evolutionary positioning," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. accepted–in press, 2014.

[45] D. Hongwei, Z. Dongfeng, and Z. Yifan, "Performance analysis of wireless sensor networks of serial transmission mode with vacation on fire prevention," *ICCET'10 IEEE CPS*, pp. 153–155, 2010.

[46] V. Mancuso and S. Alouf, "Analysis of power saving with continuous connectivity," *Computer Networks*, vol. 56, no. 10, pp. 2481–2493, 2012.

[47] J. Kennedy and R. C. Eberhard, "Particie swarm optimization," in *IEEE International Conference on Neural Networks*. Piscataway, NJ: IEEE, 1995, pp. 1942–1948.

[48] M. Hu, T. Wu, and J. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 705–720, 2013.

[49] S. Koziel and X. Yang, *Computational Optimization, Methods and Algorithms*. Berlin, Heidelberg: Springer, 2011.

[50] M. Gabryel, M. Woźniak, and R. K. Nowicki, "Creating learning sets for control systems using an evolutionary method," *Lecture Notes in Computer Science - ICAISC'2012*, vol. 7269, pp. 206–213, 2012.

[51] X. Yang, *Engineering Optimisation: An Introduction with Metaheuristic Applications*. USA: John Wiley & Sons, 2010.

[52] J. Bansal and K. Deep, "Optimisation of directional overcurrent relay times by particle swarm optimisation," in *SIS'2008 Proceedings*. IEEE, 2008, pp. 1–7.

[53] E. Baonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[54] V. Gazi and K. Passino, *Swarm stability and optimization*. Berlin, Heidelberg: Springer, 2011.

[55] X. Yang, Z. Cui, R. Xiao, A. Gandomi, and M. Karamanoglu, *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*. OXFORD, United Kingdom: Elsevier, 2013.

[56] M. Clerc and J. Kennedy, "The particle swarmexplosion, stability and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.