ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 8 No. 2, 2023

CLOUD COMPUTING WITH RESOURCE ALLOCATION BASED ON ANT COLONY OPTIMIZATION

Taras Kniazhyk, Oleksandr Muliarevych

Lviv Polytechnic National University, 12, S. Bandery str., Lviv, 79013, Ukraine
Authors' e-mail*: taras.kniazhyk.mkiks.2022@lpnu.ua*

https://doi.org/10.23939/acps2023.02.104

Submitted on 01.01.2023

© Kniazhyk T., Muliarevych O., 2023

***Abstract*: In this study, we explore the intricacies of cloud computing technologies, with an emphasis on the challenges and concerns pertinent to resource allocation. Three optimization techniques – Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Genetic Algorithm (GA) – have been meticulously analyzed concerning their applications, objectives, and operational methodologies. The study underscores these algorithms' pivotal role in enhancing cloud resource optimization, while also elucidating their respective merits and limitations.**

As the complexity of cloud computing escalates, devising efficacious strategies for resource management and allocation becomes imperative. Such strategies are paramount in aiding organizations in cost containment and performance amplification. The ensuing comparative analysis has been crafted to offer a holistic insight into the three algorithms, thus empowering cloud providers to judiciously select an optimization technique that aligns with the unique demands and challenges of their cloud computing infrastructure.

***Index Terms*: resource allocation; ACO; PSA; GA; cloud computing.**

I. INTRODUCTION

Cloud computing has surfaced as a pivotal technological advancement, furnishing businesses with scalable and economically viable solutions, a development that Dewangan et al. extensively discussed 0. Yet, proficient management of resources and cost containment in cloud ecosystems remains a significant challenge [2]. Contemporary optimization techniques, including Particle Swarm Optimization (PSO) [3], Ant Colony Optimization (ACO) [4], and Genetic Algorithms (GA) [5], are tailored to address these challenges.

The intricacies of deciphering and regulating cloud-related expenditures, especially in modern Industry 4.0 contexts, pose substantial challenges for organizations 0. Unexpected cost surges, further explored by Sehgal et al., can amplify these challenges [2]. Raj et al. emphasize the importance of designing cloud-native applications to adapt to these dynamically changing cloud environments [6].

Despite the scalability merits proffered by cloud service providers, without adept tools and services, expenditures can escalate uncontrollably [7]. A notable 73 % of cloud-centric decision-makers grapple with challenges related to optimal resource allocation, highlighting the essential role of cloud optimization 0. Singh et al. further elaborated on the dynamics of resource allocation across various computing paradigms [7].

This treatise delves into the nuances of cloud optimization, informed by the algorithms explored by Kochenderfer and Wheeler [8]. The discourse examines the significance, illustrative scenarios, and tools for fiscal management in cloud operations, aided by insights from evolutionary optimization algorithms [9].

In the domain of DevOps, the essence of cloud optimization lies in adeptly allocating resources across diverse application scenarios in our previous works [10]. However, "cloud optimization" interpretations vary among enterprises, depending on their application landscapes, a robust cloud optimization blueprint can identify avenues for refinement [4].

For entities with established cloud footprints, terms like "cloud management" often feature prominently. The insights into security concepts and practices further emphasize the need for secure resource allocation in the cloud [2].

Studies such as those by Jun et al. have revealed the potential of nature-inspired algorithms like ACO in optimizing cloud resources [11]. Meanwhile, Arianyan et al.'s focus on efficient resource allocation through genetic algorithms provides another perspective on the topic [5]. The versatility and adaptability of such algorithms in various application contexts are further echoed in works by Liu and Li [12] and Yichen et al. [13].

During the research investigation, AWS cloud infrastructure was used, despite the variety of available approaches [14] we used the same single instances [15] to exclude the influence of infrastructure implementation of Kubernetes cluster or serverless functions.

Finally, the importance of location in cloud optimization, as presented by Ekwe-Ekwe and Barker, cannot be understated, especially considering the variability in cloud resource pricing across regions [16]. And, as highlighted by Ardagna and Pernici, there's an urgent need to guarantee both global and local QoS constraints in web service selection, underlining the complexity of the resource allocation task [17]. One of the highly resource-consuming tasks, that we have experienced in the past, is image processing [18], especially during the training phase.

II. RESOURCE ALLOCATION OVERVIEW

Cloud computing refers to the delivery of hosted services over the Internet and is categorized into three types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [6]. Clouds can be public or private, to provide easy, scalable access to computing resources and IT services.

Cloud infrastructure consists of hardware and software components needed for implementing a cloud computing model. Three general service delivery categories include IaaS, PaaS, and SaaS.

IaaS: Providers like Amazon Web Services offer virtual server instances, storage, and APIs for workload migration to a virtual machine. IaaS is like a remote data center for business users.

PaaS: Cloud providers host development tools on their infrastructure, accessible through APIs, web portals, or gateway software. Examples include Salesforce's Lightning Platform and Google App Engine.

SaaS: A distribution model that delivers software applications over the internet, accessible from any location using a computer or mobile device. Examples include Microsoft 365 for productivity and email services.

Cloud computing offers several benefits such as cost reduction, pay-per-use, broad network access, multitenancy, resource pooling, and reliability. Cloud providers continually expand their services, resulting in greater usability, reliability, and scalability compared to traditional compute and storage instances. Serverless computing, for example, allows developers to create code executed by the cloud provider in response to events, eliminating the need to manage servers or instances.

Cloud computing offers businesses and clients an on-demand computing environment for leasing resources. The primary goal for both cloud consumers and suppliers is to allocate cloud resources effectively and profitably [2]. These resources are often limited, and providers must distribute them within the cloud environment's constraints while meeting the needs of cloud applications.

Resource allocation is a critical component of cloud computing, as its efficiency directly impacts the performance of the entire cloud environment [7]. Challenges in resource allocation include cost efficiency, response time, reallocation, computational performance, and task scheduling. Cloud computing consumers aim to complete tasks at the lowest possible cost. Resource allocation entails providing services and storage space for specific tasks requested by users, utilizing various allocation strategies. These strategies involve coordinating cloud provider activities when allocating scarce cloud resources and addressing cloud applications' requirements.

Cloud users and providers, the two parties in a cloud computing environment, have distinct objectives. Providers seek to maximize resource utilization and profits, while users aim to reduce expenses without compromising performance. Various strategies help maintain a balance between resource allocation and cost, preventing over-provisioning, under-provisioning, resource fragmentation, and resource contention.

Resource allocation in the cloud has some limitations:

1) cloud users rent resources from remote servers but lack control over them, making migration and searching for better data storage challenging;
2) data transfer between providers can pose risks, with data stored in the public cloud being vulnerable to phishing and hacking attacks. Malware can spread rapidly due to interconnected servers;
3) in-depth knowledge is necessary for mastering resource allocation and management since cloud service providers hold extensive information about cloud operations;
4) local software installation is needed for using peripheral devices like scanners and printers with the cloud, though networked peripherals work well with the cloud.

Enterprises continue to rely on cloud computing for various use cases, including increasing efficiency, optimizing costs, ensuring data security, and storing unlimited data. Knowledge of resource allocation is increasingly essential for cloud users seeking to optimize their cloud usage costs. While many strategies exist for allocating resources in the cloud, the most effective method is the one that satisfies cloud users and maximizes income for cloud service providers.

III. PURPOSE OF WORK

This research seeks to meticulously formulate a proficient resource allocation blueprint that leverages the capabilities of the Ant Colony Optimization (ACO) algorithm, with a distinct aim to enhance the time efficiency in resource allocation within cloud computing environments.

The experimental resource allocation system, based on classical methods, can handle 100 tasks in nearly 1000 ms. This data was derived from a series of preliminary experiments conducted in a controlled AWS environment. The specific cloud computing setup was replicated using methology and details described by Dewangan et al. in their investigations [1]. The aim is to find a nature-inspired method that has better performance, with at least a 5 % calculation time decrease.

The comparative effectiveness of the ACO, PSO, and GA algorithms relative to classical methods, specifically in the realm of optimization tasks and resource allocation in the cloud computing, is detailed in the distinct works [8, 9, 11].

The main purpose of current article is to conduct a comprehensive comparative study juxtaposing ACO against PSO and GA in terms of foundational mechanisms, representative tasks, and applications in cloud computing, aiming to underline the substantial and quantifiable benefits of ACO in specified cloud computing scenarios.

Investigate the foundational mechanisms of ACO, PSO, and GA, unraveling their operating principles and

potential bottlenecks in the context of cloud computing applications. Adjust and optimize the parameters of the ACO algorithm to enhance its performance against the set metrics. Finally, Performance Testing – to evaluate and define the most optimal nature-inspired method under simulated cloud computing environments and specific task loads (like 100 tasks and more), ensuring the rigorous assessment of its efficacy and efficiency in diverse scenarios.

As a summary step, perform an in-depth analysis of the resulting data, with particular attention to the computational time of the ACO, PSO, and GA algorithms, further refining the algorithm to bolster its efficiency and effectiveness.

IV. ADJUSTMENT OF NATURE-INSPIRED OPTIMIZATION METHODS

Many local optimization algorithms are gradient-based. As indicated by their name, these methods leverage gradient information to pinpoint an optimal solution. Owing to their efficiency (in terms of the number of function evaluations required to determine the optimum), the ability to address problems with multiple design variables, and a minimal need for problem-specific parameter adjustments, gradient-based algorithms have found extensive applications in diverse engineering optimization problems. Yet, they come with their set of limitations: propensity to identify only local optima, challenges in handling discrete optimization problems, intricate implementation, and a vulnerability to numerical disturbances. In essence, these algorithms have been thoroughly researched and are well-documented [8].

An exploration of local search methods wouldn't be comprehensive without acknowledging non-gradient-based local search techniques such as Powell's method and the Nelder-Mead simplex algorithm. Both can manage non-linear, unconstrained optimization challenges. While Powell's method hinges on the principle of conjugate directions, the Nelder-Mead approach uses a simplex coupled with a series of basic rules to mirror the least favorable vertex through the centroid of the simplex.

Over recent decades, evolutionary optimization algorithms [9] have ascended in prominence. Contrary to local techniques that modify a singular design point (often influenced by gradient data) in successive iterations, these algorithms forgo gradient details, typically relying on a collection of design points (or a "population") to discern the optimal design. Drawing inspiration from natural processes, these methodologies offer strengths like robustness, a heightened chance of uncovering a global or near-global optimum, straightforward implementation, and adaptability to discrete optimization challenges. Nonetheless, they aren't without their challenges: pronounced computational demands, subpar constraint management, the necessity for problem-specific parameter adjustments, and constraints on problem dimensions. Now, let's delve into several nature-inspired algorithms evaluated within the framework of the developed computer system: PSO, ACO, and GA.

A. PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle Swarm Optimization (PSO) is a nature-inspired algorithm presenting an uncomplicated mechanism to scout for an optimal solution within a designated solution domain. What sets PSO apart from myriad other optimization techniques is its exclusive dependency on the objective function, without any reliance on the gradient or any differential form of the objective. Moreover, its hyperparameters are relatively limited [3].

To mathematically model the PSO principles and enable the swarm to find the global minimum of a fitness function we need to consider the following:

- Each particle in Particle Swarm Optimization has an associated position, velocity, and fitness value.
- Each particle keeps track of its "particle best-Fitness value" and "particle bestFitness position".
- A record of "global bestFitness position" and "global bestFitness value" is maintained.

Fig. 1. represents the data structure used to store the swarm population. This data structure holds information about each particle's position, velocity, and fitness value, as well as their best individual and global positions.

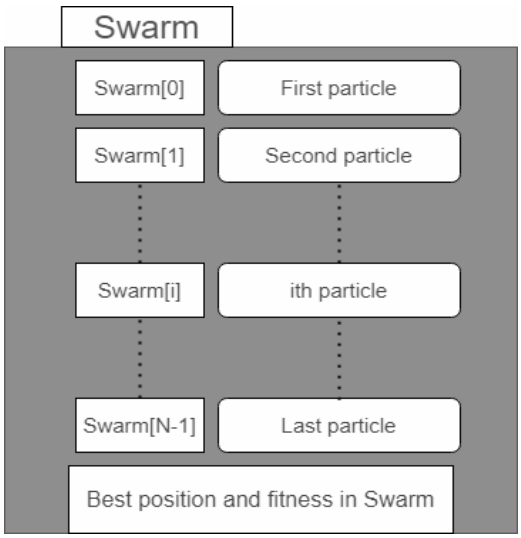

Fig. 1. Data structure to store Swarm population

Fig. 2. illustrates the data structure used to store the i*th* particle of the swarm.

This data structure captures the details specific to the individual particle, such as its position, velocity, and fitness value. Here is the proposed approach used to adjust PSO for resource allocation problem-solving.

Step 1. Initialization:

- Define the number of particles in the swarm *N*.
- For each particle i, randomly initialize the position representing a potential solution to resource allocation.
- The position vector might represent how different resources (CPU, memory) are allocated to tasks. Initialize the velocity for each particle.
- Evaluate the fitness of each particle using a predefined objective function – the cost of resources.

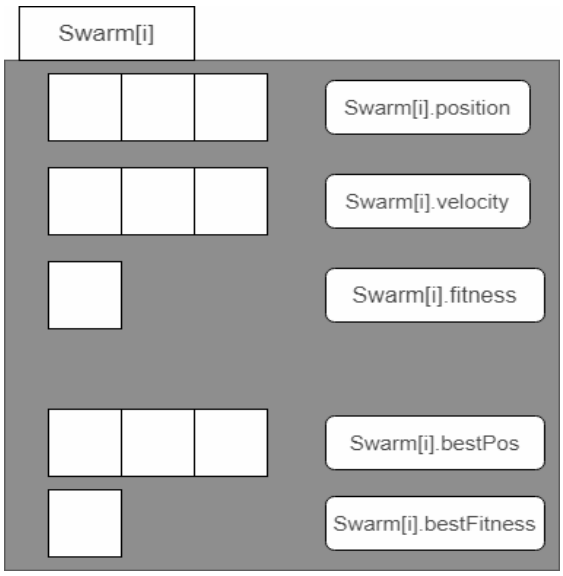

Fig. 2. Data structure to store ith particle of Swarm

Step 2. Iterative Optimization running in a two-level cycle with max iteration criteria and per each particle:

- Update the personal best position (pBest) if the current position has better fitness and update the global best position (gBest) among all particles.
- Update the velocity of the particle.
- Update the position of the particle.
- If the new position is better than the pBest, update the pBest.
- Update the global best position (gBest) among all particles.

Step 3. Post Optimization: The gBest position represents the optimized resource allocation solution, using it we apply result allocation in the cloud environment. Try to evaluate current resources and consider dynamic changes.

A remarkable aspect of PSO is the presence of a stable topology, where particles can communicate with one another and enhance the learning rate to achieve the global optimum. The metaheuristic nature of this optimization algorithm offers numerous opportunities, as it iteratively seeks to improve a candidate solution by optimizing the problem at hand.

B. ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) [10] stands out as a stochastic method devised to trace optimal routes, inspired by the behavior of ants foraging for the most efficient path between their colony and food sources. Originally conceptualized for the traveling salesman problem, ACO's applicability has extended to a variety of intricate optimization challenges [4]. Ants, being social insects, communicate through pheromone trails. Elevated pheromone concentrations enhance the likelihood of a path's selection by subsequent ants [11]. To grasp ACO's application to the traveling salesman problem (TSP) [10], one should visualize a fully integrated construction graph where vertices denote cities and edge lengths correlate to inter-city distances. Pheromone concentrations and heuristic metrics are tethered to graph edges. Ants devise solutions by probabilistically selecting edges, guided by these metrics. Once all ants complete their tours, pheromone concentrations are revised. This process iterates until a termination criterion is satisfied. Within the realm of resource allocation, ACO excels in navigating through graphs representing myriad resource allocation alternatives.

Here is the proposed approach used to adjust ACO for the resource allocation problem-solving.

Step 1. Initialization – define the number of ants in the colony N. Place the ants in random positions. Initialize the pheromone levels on the paths.

Step 2. Construct Ant Solutions running in cycle two procedures:

- Construct a solution by moving the ant from state to state within the feasible solution space.
- At each step, apply a stochastic local search procedure biased by pheromone levels to simulate decision-making.

Step 3. Pheromone Update:

- Evaporate some portion of the pheromone on all paths.
- For each ant, deposit pheromone on the paths it took based on the quality of its solution.

Step 4. Daemon Actions – this phase evaluates currently available resources and considers dynamic changes.

Step 5. Convergence Check – check for convergence criteria. If not converged, go to Step 2.

Step 6. Solution Construction – construct the resource allocation solution from the highest pheromone path.

C. GENETIC ALGORITHM

Define Genetic algorithm (GA) [5] is an optimization method based on natural selection that solves constrained and unconstrained problems. GA modifies a population of individual solutions over successive generations, evolving toward an optimal solution [12].

Three main rules [13] are used in GA to create the next generation: selection rules (parents are stochastically selected based on their fitness scores); crossover rules (two parents are combined to form children); mutation rules (random changes are applied to individual parents to form children).

Here is the proposed approach used to adjust GA for resource allocation problem-solving.

Step 1. Initialization – generate an initial population of chromosomes, where each chromosome is a feasible resource allocation. Evaluate the fitness of each chromosome in the population.

Step 2. Selection – select chromosomes to form a new population; selection should favor fitter solutions.

Step 3. Crossover – with a crossover probability, crossover the pairs of chromosomes randomly chosen from the population. This is done to reproduce a new population of chromosomes.

Step 4. Mutation – with mutation probability, mutate the new population at each position.

Step 5. Evaluation – evaluate the individual fitness of each chromosome in the current population.

Step 6. Termination Check – if a maximum number of generations has been reached or dynamic changes appeared, we need to terminate. If the result is invalid or the calculation is not finished, go back to Step 2.

Step 7. Solution Construction – the best chromosome represents the optimized resource allocation solution, apply this allocation in the cloud environment.

Genetic algorithms possess inherent parallel processing capabilities, making them apt for optimizing a diverse array of challenges, from discrete functions to multi-objective problems. Their operation is not contingent upon derivative data, and they are equipped to yield multiple optimal outcomes. However, the traditional GA isn't optimized for simple challenges and might encounter computational hurdles due to recurrent fitness value evaluations.

V. CASE STUDY

In this section, we provide an empirical assessment of three optimization algorithms previously described: Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA), all tailored to tackle resource allocation issues within cloud computing environments. These algorithms' performances were juxtaposed within a rigorously designed experimental framework.

In Fig. 3. there are showcases a comparative performance assessment of Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) as they navigate resource allocation challenges in cloud computing settings.

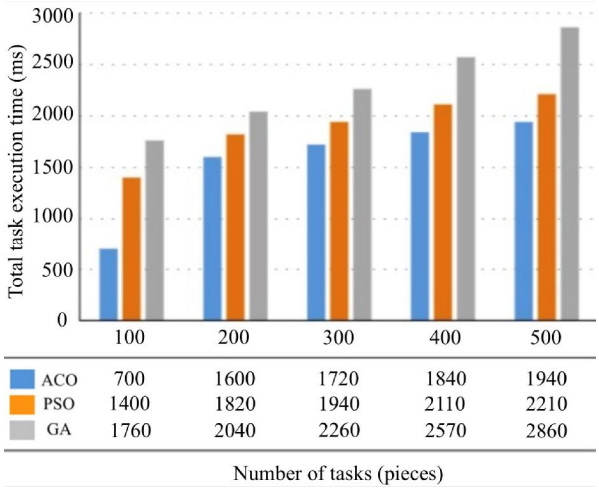

Fig. 3. Performance test results of ACO, PSO and GA algorithms

The evaluations were orchestrated within a virtual cloud setting facilitated by Amazon Web Services (AWS) [14]. This cloud infrastructure was composed of a network of EC2 instances, comprising a diverse blend of compute-optimized (c5.large) and memory-optimized (r5.large) instances. Such a blend epitomizes the heterogeneity commonplace in contemporary cloud platforms [15]. Notably, every c5.large instance was outfitted with 2 vCPUs and 4 GB of RAM, whereas each r5.large counterpart was provisioned with 2 vCPUs and 16 GB of RAM.

The experimental workload encompassed a spectrum of computational tasks that are archetypal of cloud operations, such as data processing and graphical data storage processing. These tasks came with their unique resource prerequisites and QoS stipulations [17].

Our prime objective was twofold: curtailing both the makespan (cumulative completion timeframe) and the associated costs while guaranteeing a foundational performance threshold for each task. As a result, a multi-objective fitness function was deemed apt for assessments.

Algorithm test configurations for Ant Colony Optimization (ACO):

- number of ants is 20;
- the evaporation rate is 0.1;
- pheromone influence is 1.0;
- heuristic influence is 1.0.

Algorithm test configurations for PSO:

- number of particles is 40;
- inertia weight is 0.729;
- the cognitive constant is 1.49445;
- social constant is 1.49445.

Algorithm test configurations for GA:

- population size is 50;
- crossover probability is 0.8;
- mutation probability is 0.02.

Each algorithm underwent 30 autonomous runs. During each iteration, the algorithm endeavored to allocate resources to a task assortment. The tasks and their specifications were crafted from a blend of authentic traces coupled with synthetic workloads, ensuring a comprehensive representation of diverse scenarios.

The tasks integrated within this experimental framework spanned an array of computational endeavors, encompassing data processing, machine learning model development, scientific simulations, and expansive optimization challenges. Given the hefty computational demands and distinct requisites of these tasks, adept resource allocation emerges as indispensable for streamlined operations, user gratification, and peak resource deployment.

By analyzing the experimental results (see Fig. 3.), it is evident that the ACO algorithm outperforms PSO and GA in the context of the testing tasks. The ACO compared to PSO for 100 tasks requires less than 50 % of the calculation time (700 ms per ACO and 1400 ms per PSA). The ACO compared to GA for 100 tasks requires less than nearly 60 % of the calculation time (700 ms per ACO and 1760 ms per PSA). Moreover,

when the number of tasks increases up to 500, ACO still keeps its performance benefit compared to other tested nature-inspired methods.

The developed system using adjusted ACO resource allocation can compute 100 tasks on average for 700 ms, compared to nearly 1000 ms per experimental system using the classical method application. It means we got a boost of the performance by more than 30 % when the target expected threshold for nature-inspired methods was at least 5%.

VI. CONCLUSION

The article demonstrated how nature-inspired optimization methods could be adjusted to solve resource allocation problems. Our research established the effectiveness of the Ant Colony Optimization algorithm for resource allocation in cloud computing environments. Despite its computational expense and parameter tuning requirements, ACO delivered superior performance compared to PSO and GA. Future work includes refining the algorithm to minimize its disadvantages and exploring parallelization techniques to enhance real-time application.

The proposed ACO-based method for cloud computing resource allocation has the potential to cater to a wide range of tasks while maintaining efficiency, flexibility, and adaptability in dynamic and complex cloud computing environments. The improvement of the developed system with adjusted ACO resource allocation implementation increases the performance of task processing by more than 30 % better than the experimental system (700 ms per ACO and nearly 1000 ms per experimental system using the classical approach for the 100 tasks).

The methodology of how nature-inspired methods, particularly Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA), could be used for resource allocation problems offers a robust framework for further research in this scientific area. Future studies could also explore the algorithm's interoperability across diverse cloud service models, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), thereby amplifying its applicability and utility in various cloud computing facets and ensuring that the methodologies developed are robust, versatile, and applicable across a multitude of scenarios and applications.

REFERENCES

[1] Dewangan B., Choudhury T., Toe T., Singh B., Nhu N., Tomar R. (2021). Cloud Autonomic Computing in Cloud Resource Management in Industry 4.0. Switzerland: Springer, pp. 123–195. DOI: https://doi=10.1007/978-3-030-71756-8_9

[2] Sehgal N., Bhatt. P., Acken J. (2020). Cloud Computing with Security, Concepts and Practices. Second edition. Switzerland: Springer, pp. 75–109. DOI: https://doi=10.1007/978-3-030-24612-9

[3] Cai J., Peng P., Huang X. and Xu B. (2020). A Hybrid Multi-Phased Particle Swarm Optimization with Sub Swarms, *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, Beijing, China, pp. 104–108. DOI: https://doi=10.1109/ICAICE51518.2020.00026

[4] Kozlov O. (2021). Information Technology for Designing Rule bases of Fuzzy Systems using Ant Colony Optimization, *International Journal of Computing*, 20(4), pp. 471–486. DOI: https://doi.org/10.47839/ijc.20.4.2434

[5] Arianyan E., Maleki D., Yari A. and Arianyan I. (2012). Efficient resource allocation in cloud data centers through genetic algorithm, *6th International Symposium on Telecommunications (IST)*, Tehran, Iran, pp. 566–570. DOI: https://doi=10.1109/ISTEL.2012.6483053

[6] Raj P., Vanga S., Chaudhary A. (2022). Cloud-native Computing: How to Design, Develop, and Secure Microservices and Event-Driven Applications, John Wiley & Sons, pp. 129–163. DOI: https://doi.org/10.1002/9781119814795.ch13

[7] Singh A., Indrusiak L., Dziurzanski P. (2022). Dynamic Resource Allocation in Embedded, High-Performance and Cloud Computing, e-book, Denmark: River Publishers, pp. 128–154. [Electronic resource]. Available at: https://eprints.whiterose.ac.uk/106984/1/Published_ebook_RP_E9788793519077.pdf (Accessed: 01 January 2023)

[8] Kochenderfer M. J., Wheeler T. A. (2019). Algorithms for Optimization. United Kingdom: MIT Press, pp. 125–189. DOI: https://doi.org/10.1109/MCS.2019.2961589

[9] Badar, Altaf Q. H. (2021). Evolutionary Optimization Algorithms. United States, CRC Press, pp. 113–218. DOI: https://doi.org/10.1201/b22647

[10] Muliarevych O. (2016). Solving dynamic assymetrical Travelling Salesman Problem in conditions of partly unknown data, Lviv-Slavsk, Lviv Polytechnic Publ., *TCSET'2016* vol.1, pp. 446–448. DOI: https://doi.org/10.1109/TCSET.2016. 7452084

[11] Jun S., Yatskiv N., Sachenko A. and Yatskiv V. (2012). Improved method of ant colonies to search independent data transmission routes in WSN, *2012 IEEE 1st International Symposium on Wireless Systems (IDAACS-SWS)*, Offenburg, Germany, pp. 52–57. DOI: https://doi.org/10.1109/IDAACS-SWS.2012.6377632

[12] Liu S., Li Z. (2017). A modified genetic algorithm for community detection in complex networks, *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, Chennai, India, pp. 1–3. DOI: https://doi.org/10.1109/ICAMMAET.2017.8186747

[13] Yichen L., Bo L., Chenqian Z. and Teng M. (2020). Intelligent Frequency Assignment Algorithm Based on Hybrid Genetic Algorithm, *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, Chongqing, China, pp. 461–467. DOI: https://doi.org/10.1109/CVIDL51233.2020.00-50

[14] Muliarevych O. (2022), Acceptance and shipping warehouse zones calculation using serverless approach, *12th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Athens, Greece, pp. 1–6. DOI: https://doi.org/10.1109/DESSERT58054.2022.10018786

[15] Sampaio A. M. and Barbosa J. G. (2019). Enhancing Reliability of Compute Environments on Amazon EC2 Spot Instances, *2019 International Conference on High Performance*

Computing & Simulation (HPCS), Dublin, Ireland, pp. 708–715. DOI: https://doi.org/10.1109/HPCS48598.2019.9188116

[16] Ekwe-Ekwe N. and Barker A. (2018). Location, Location, Location: Exploring Amazon EC2 Spot Instance Pricing Across Geographical Regions, *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Washington, DC, USA, pp. 370–373. DOI: https://doi.org/10.1109/CCGRID.2018.00059

[17] Ardagna D. and Pernici B. (2005). Global and local QoS constraints guarantee in Web service selection, *IEEE International Conference on Web Services (ICWS'05)*, Orlando, FL, USA, pp. 805–806. DOI: https://doi.org/10.1109/ICWS.2005.66

[18] Tsiunyk B., Muliarevych O. (2022). Software System for Motion Detection and Tracking, *Advances in Cyber-Physical Systems*, 7(2), pp. 156–162. DOI: https://doi.org/10.23939/ acps2022.02.156

Taras Kniazhyk received a Bachelor's degree in Computer Engineering at Lviv Polytechnic National University in 2019. Since 2022 he has been receiving a Master's degree.

Oleksandr Muliarevych is an Associate Professor at the Computer Engineering Department at Lviv Polytechnic National University.

He earned his PhD Degree in Computer Systems and Components at Lviv Polytechnic National University in 2016.

Research areas: Multi-Agent Systems, Swarm Intelligence, Computer Vision, Internet of Things, Decentralized Cyber-Physical Systems.