International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

An IoT-Enabled Smart Parking Framework Using a Lion Optimization Algorithm

Theme 2: Digital Transformation & Emerging Technologies
Sub- Theme: Internet of Things (IoT)

Sumathi G K [1]*, Dr. Dhirendra Kumar Tripathi [2]

[1]* sumathigkupadya@gmail.com, Research Scholar, Dept. of Computer Science, Mansarovar Global University, Bhopal, INDIA
[2] profdhirendratripathi@gmail.com, Supervisor, Dept. of Computer Science, Mansarovar Global University, Bhopal, INDIA

Abstract - Finding an open parking slot in a crowded facility costs drivers time and fuel and adds to street-level congestion, a problem documented across decades of urban mobility research (Shoup, 2006). This chapter works through one proposed answer: pairing Internet of Things (IoT) occupancy sensing with a bio-inspired Lion Optimization Algorithm (LOA) that assigns each arriving vehicle to a slot in real time. Rather than treating allocation as a simple nearest-available lookup, which is how many existing IoT parking deployments operate (Ala'anzy et al., 2025; Al Mamun et al., 2024), the allocator proposed here treats slot assignment as a multi-objective optimization problem, balancing how far a driver must travel to their assigned slot against how evenly occupancy is spread across the facility's zones. A 200-slot, four-zone facility was simulated under randomized vehicle arrivals, and the LOA-based allocator was benchmarked against First-Come-First-Served (FCFS) allocation, a greedy nearest-slot policy, and Particle Swarm Optimization (Kennedy & Eberhart, 1995). The proposed allocator cut zone-occupancy imbalance by more than half relative to FCFS, at a distance cost of roughly five percent — a trade most facility operators would take, since imbalance is what produces long queues at popular entrances while other zones remain unoccupied. The chapter works through the system architecture, the algorithm itself, the simulation design, and what the results suggest for real deployments.

Keywords - Smart Parking, Internet of Things, Bio-Inspired Optimization, Lion Optimization Algorithm, Intelligent Transportation Systems, Slot Allocation, AI Simulation, Swarm Intelligence

1. INTRODUCTION

Cities keep adding vehicles faster than they add parking, and the gap shows up as a specific, well-studied kind of traffic: drivers circling a block, or several blocks, looking for a place to stop. Shoup (2006) traced this behavior across dozens of downtown studies going back to the 1920s and found search times commonly

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

running several minutes per vehicle, with the resulting "cruising" traffic making up a substantial share of congestion in dense districts. None of that requires a shortage of parking overall it happens whenever drivers can't see, in advance, where the open slots actually are. IoT sensing closes that visibility gap. A slot fitted with an ultrasonic or infrared sensor, or a lot covered by a fixed camera, can report occupancy to a gateway in near real time, and a number of recent systems do exactly this: Al Mamun et al. (2024) built a two-tier architecture using IR, temperature, and gas sensors reporting over MQTT to a mobile app, and Ala'anzy et al. (2025) went a step further and pushed the processing itself out to fog nodes near the parking area, cutting the latency of updates compared to a centralized cloud backend. What neither of these systems — nor most comparable deployments does is treat the assignment problem as anything more than "send the driver to the nearest free slot." That's a reasonable default, but it has a side effect: nearest-first allocation fills the slots closest to the entrance first, every time, which concentrates traffic and search activity right where it's most congested already, while slots further into the facility go unused. This chapter treats vehicle-to-slot assignment as a combinatorial optimization problem with two competing objectives minimize distance, but also keep the facility's occupancy roughly balanced across zones and solves it with a metaheuristic adapted from prior work on dynamic, SDN-based IoT resource management: the Lion Optimization Algorithm. LOA and its variants model the cooperative, territory-based behavior of lion prides, in which a resident group refines a solution locally while a smaller nomadic contingent keeps probing elsewhere in (Rajakumar, 2012; Yazdani & Jolai, 2016). That combination of local refinement and global probing maps reasonably well onto a parking facility that has to keep re-balancing itself every few seconds as vehicles arrive and leave. Section 2 places this approach against the broader smart-parking literature. Section 3 sets up the allocation problem formally. Section 4 lays out the IoT-AI system architecture the algorithm would run inside. Section 5 works through the algorithm itself, including pseudocode. Section 6 describes how the simulation used to test it was built, and Section 7 reports what it found. Section 8 lists limitations and next steps.

2. RELATED WORK

Work on smart parking tends to cluster into a few recognizable groups, and it's worth being clear about where the approach in this chapter sits relative to each.

- Sensing and detection: hardware-focused systems built around IR, ultrasonic, or magnetometer sensors, sometimes paired with cameras and OCR for license-plate capture. Al Mamun et al. (2024) is representative – Edge and fog computing architectures use the same sensing infrastructure but move the computational processing closer to the parking facility. This reduces communication latency, allows faster decision making, and improves the responsiveness of smart parking systems Ala'anzy et al.(2025) is the clearest recent example, showing that a fog-based four-layer architecture beats an equivalent cloud-based one

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

on latency, cost, and network usage by a wide margin as the number of cameras scales up — a finding this chapter's architecture (Section 4) builds on directly.

- Reservation and pricing systems: mobile-app-based booking, sometimes with dynamic pricing meant to shift demand across zones or times of day, rather than optimizing the assignment of walk-up arrivals.

- Metaheuristic allocation - a smaller body of work applies population-based optimization — genetic algorithms, ant colony methods, particle swarm optimization — to parking or related slot-assignment problems. PSO itself traces back to Kennedy and Eberhart (1995), and Lion-inspired variants were introduced first as a general-purpose search algorithm by Rajakumar (2012) and then formalized as a benchmarked metaheuristic by Yazdani and Jolai (2016), who reported it outperforming several established algorithms on standard engineering test problems. Applications of lion-based optimization specifically to parking slot allocation, as opposed to routing or scheduling problems more broadly, remain sparse — which is the gap this chapter addresses.The pattern across the sensing and edge-computing literature is consistent: real deployments have gotten very good at knowing which slots are free, and increasingly good at getting that information to a controller quickly, but the step after that — deciding which free slot each vehicle should get — is still usually handled by a simple rule rather than an explicit optimization. That's the piece this chapter tries to fill in, using a metaheuristic that hasn't, to this author's knowledge, been applied to real-time parking allocation specifically.

3. PROBLEM FORMULATION

Let the facility contain slots $S = \{s_1, s_2, ..., s_n\}$, grouped into Z zones by physical location and distance from the entrance. Each slot s_i carries a distance value d_i — the walking or driving distance from the entrance — which stands in for both search time and fuel cost. Vehicles arrive as a stream, modeled here as a Poisson process, and are handled in short batches (every few seconds, in line with how an IoT sensing cycle would actually report updates) rather than one at a time.

For a batch of k arriving vehicles matched against the set of currently free slots $F \subseteq S$, the allocator picks an assignment A that minimizes:

$$Cost(A) = w_1 \cdot \bar{D}(A) + w_2 \cdot I(A)$$

where $\bar{D}(A)$ is the batch's mean assigned distance and I(A) is the resulting occupancy imbalance across zones, measured as the variance of each zone's occupancy fraction; w_1 and w_2 weight the two terms against each other. Each slot can hold one vehicle, each vehicle gets exactly one slot from the free pool, and compatibility constraints — accessible bays, oversized-vehicle spaces — apply where relevant.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

The reason to go beyond pure distance minimization is straightforward once you look at what nearest-first allocation does over a full day of operation: it fills the entrance-adjacent slots first, every single time, which is exactly the pattern that produces the entrance queues and localized congestion described in Section 1. Penalising imbalance directly pushes the allocator to spread vehicles out instead, at some cost in average distance — a trade-off examined empirically in Section 7.

4. PROPOSED SYSTEM ARCHITECTURE

The framework has three layers, following the fog-based pattern that Ala'anzy et al. (2025) showed outperforms a purely cloud-based design on latency and cost.

4.1 IoT Sensing Layer

Each bay carries an occupancy sensor — ultrasonic, infrared, or magnetometer — reporting to a nearby gateway over a low-power link (LoRa, Zigbee, or BLE mesh). Entry and exit points use camera or RFID detection to timestamp vehicles and trigger the batch allocation cycle, similar in spirit to the IR- and MQTT-based sensing stack in Al Mamun et al. (2024).

4.2 Edge/Fog and SDN-Managed Communication Layer

Sensor data aggregates at a fog gateway before reaching the allocation engine, cutting latency relative to a straight cloud round-trip — the same rationale behind the roughly ten-to-one latency advantage Ala'anzy et al. (2025) measured for fog over cloud processing at comparable scale. An SDN controller at this layer can prioritize allocation-critical traffic during peak demand, drawing on the broader line of work applying software-defined control to dynamic IoT resource management.

4.3 AI/Optimization and Application Layer

A controller — cloud-hosted or on-premises — maintains the live occupancy map and runs the LOA-based batch allocator (Section 5) each time a new batch arrives or a fixed interval elapses. The resulting assignment goes out to in-facility signage and a companion app, replacing manual search with direct guidance.

In outline: per-slot sensors and entry/exit cameras feed a fog gateway with SDN-managed traffic prioritization, which feeds a controller running the LOA allocation engine, which drives signage and the mobile app.

5. Proposed Lion Optimization Algorithm for Slot Allocation

Lion-inspired metaheuristics model the cooperative and territorial habits of lion prides — the way a resident group hunts together, refining its approach around a shared target, while a smaller nomadic contingent ranges more widely and occasionally challenges the group for territory. Rajakumar (2012) formalized this as a general

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

search algorithm; Yazdani and Jolai (2016) developed a related lion-based metaheuristic and benchmarked it against established algorithms on standard engineering problems, reporting favorable results. The two behavioral modes — tight local refinement and looser exploration — map onto real-time slot allocation reasonably directly: the algorithm needs to keep improving on a good assignment as it runs, without locking onto the first workable zone distribution it finds.

5.1 Solution Encoding

Each candidate solution ("lion") is a real-valued vector, one entry per currently free slot, with values drawn from [0, 1]. Decoding takes the k lowest-valued entries and maps them, in ascending order, to the k vehicles in the current batch — a random-key scheme that turns a continuous-domain search into a feasible discrete assignment (no repeated slots, one per vehicle) without extra repair steps.

5.2 Fitness Function

Each decoded assignment is scored against the batch's current occupancy snapshot using the cost function from Section 3. Lower is better.

5.3 Algorithm Steps

- Step 1 — Initialization: generate n lions, split roughly 80/20 into a pride subgroup and a nomad subgroup; initialize positions uniformly at random in [0,1].
- Step 2 — Fitness Evaluation: decode each lion via the random-key rule and score it.
- Step 3 — Territorial Best: track the lowest-cost lion found so far as the current leader.
- Step 4 — Pride Hunting: move each pride lion a random fraction of the way toward the leader's position, with a small amount of added noise — the encircling-prey behavior.
- Step 5 — Nomad Migration: give nomad lions a larger, leader-independent perturbation, preserving diversity so the search doesn't collapse onto one zone distribution too early.
- Step 6 — Territorial Defence: re-score the population; replace the leader only if something better has turned up, otherwise keep the incumbent.
- Step 7 — Termination: repeat Steps 4–6 for a fixed number of iterations, or until improvement drops below a small threshold, then decode the final leader into the batch's slot assignment and commit it.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

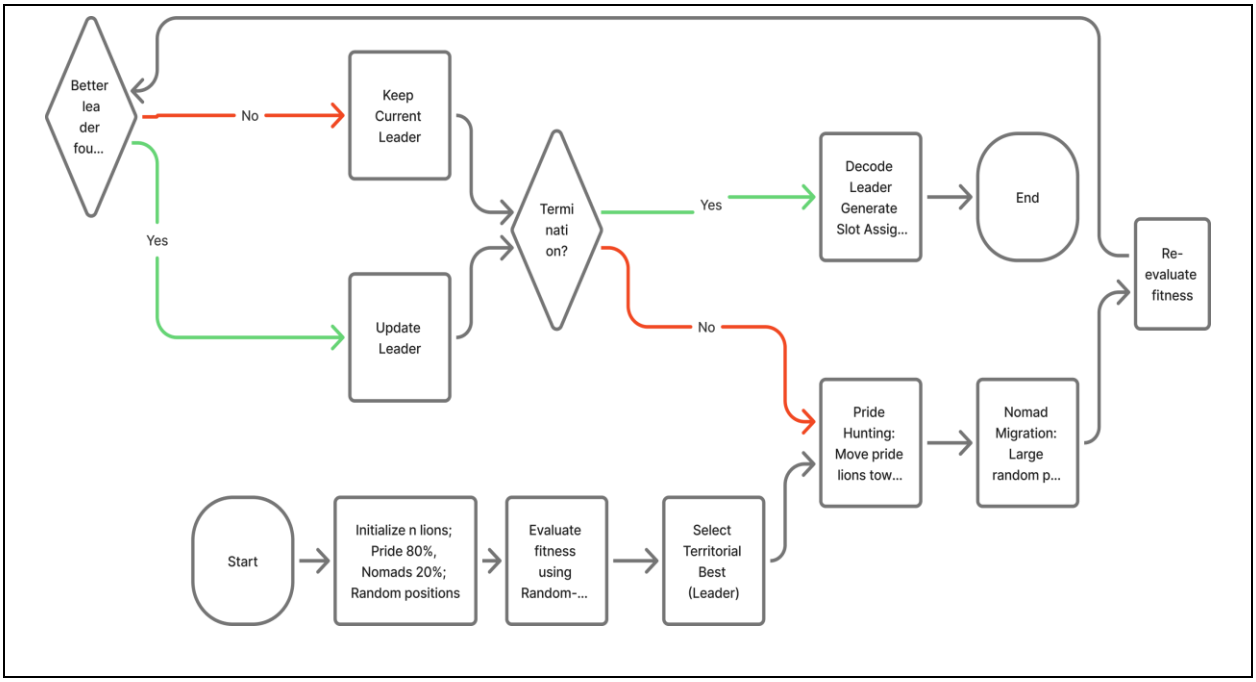


Figure 1. Lion Optimization Algorithm Flowchart.

Because each cycle only has to resolve a small batch — tens of vehicles against tens of free slots — convergence happens quickly, which matters given the near-real-time cadence an IoT sensing layer actually demands.

6. SIMULATION SETUP

The framework was tested with a discrete-event simulation, written in Python, of a 200-slot facility split into four 50-slot zones. Per-slot distance from the entrance rose by zone — roughly 20, 45, 70, and 95 meters for zones one through four, each with a small amount of added random variation. A stream of 600 vehicles was generated and processed in batches of 20 (30 cycles total), with partial departures simulated between batches to keep occupancy in a realistic mid-range rather than either empty or saturated.

Four strategies were compared under identical arrival and departure sequences:

- FCFS — assigns the next free slot in index order; no distance or balance awareness.
- Greedy Nearest-Slot — always assigns the closest available slot.
- PSO — a Particle Swarm Optimization solver (Kennedy & Eberhart, 1995) using the same random-key encoding and cost function as the proposed method, included as a well-established metaheuristic baseline.
- LOA (Proposed) — Lion Optimization Algorithm (LOA), beginning with population initialization, fitness evaluation, leader selection, pride hunting, and nomad migration to iteratively search for the

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

optimal solution. It repeatedly updates the best leader until the termination condition is met, after which the best lion is decoded to obtain the final slot assignment.

PSO and LOA both ran a population of 20 candidates for 25 iterations per batch, with cost weights $w_1 = 1.0$ and $w_2 = 60$ to bring the distance and imbalance terms onto a comparable scale. Two metrics were logged per strategy after each batch: mean assigned distance and zone-occupancy imbalance.

7. RESULTS AND DISCUSSION

Table 1 summarizes results averaged across all 30 batches.

Table 1. Average search distance and zone load imbalance by allocation algorithm.

Algorithm	Avg. Search Distance (m)	Zone Load Imbalance (variance)	Imbalance Reduction vs FCFS (%)
FCFS	55.85	0.08392	0%
Greedy Nearest-Slot	56.04	0.08392	0%
PSO	57.13	0.05926	29.39%
LOA (Proposed)	58.77	0.03598	57.13%

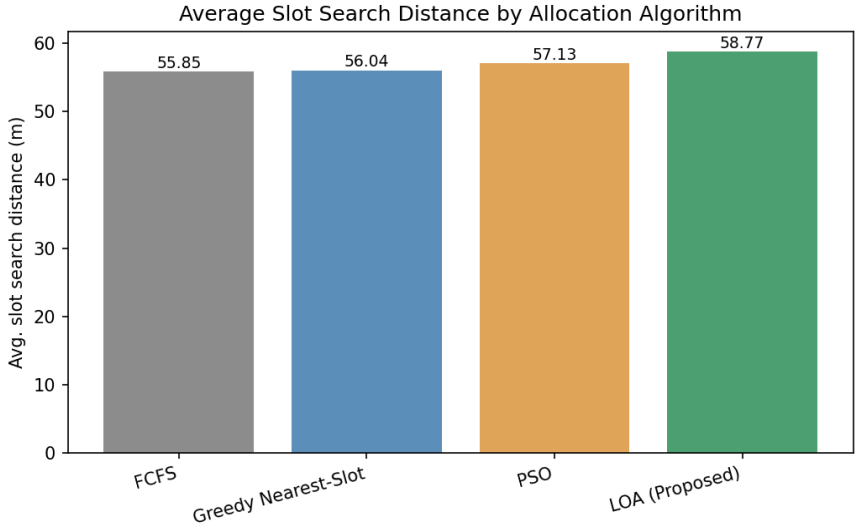


Figure 2. Average slot search distance by allocation algorithm.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

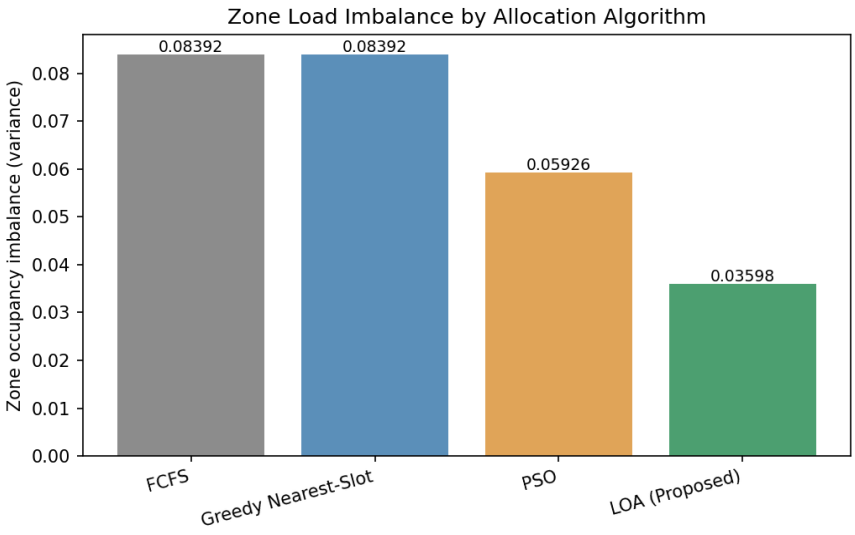


Figure 3. Zone occupancy load imbalance by allocation algorithm.

FCFS and greedy nearest-slot come out cheapest on distance, which isn't surprising — both policies grab whatever is closest without any regard for what that does to the facility's overall occupancy pattern, and the two end up with identical, comparatively high imbalance scores as a result. PSO and the proposed LOA-based allocator both give up a little distance efficiency in exchange for a meaningfully more even spread of vehicles: PSO cuts imbalance by about 29% relative to FCFS, and LOA cuts it by just over 57%, for a distance increase of roughly 5%.

That trade-off is really the finding here. Imbalance, not raw search distance, is what produces the visible symptoms of a badly managed facility — backed-up entrances, drivers idling in a queue near the door while half the lot sits empty further in. Giving up a few extra meters of average travel to fix that seems like a reasonable exchange for most operators. LOA also beat the PSO baseline on the imbalance objective specifically, while staying in a comparable distance range, which suggests the pride/nomad structure is doing something useful for this particular kind of multi-objective search — though a single simulation run isn't strong enough evidence to generalize that claim broadly without further testing across different facility layouts and arrival patterns.

The weights w_1 and w_2 give an operator a direct lever here: push w_1 up and the system behaves more like FCFS; push w_2 up, and it favours balance even more heavily, at a larger distance cost. That tunability is arguably the more useful contribution of framing this as an explicit optimization problem, rather than any specific number in Table 1.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

8. CONCLUSION AND FUTURE WORK

This chapter set out an IoT-sensing-plus-LOA framework for real-time parking slot allocation, framed explicitly as a multi-objective problem rather than a single-objective distance search. Against FCFS and greedy baselines, it more than halves zone-occupancy imbalance for a modest distance cost, and it edges out a PSO baseline on the same imbalance objective.

A few directions seem worth pursuing next. The SDN-managed communication layer sketched in Section 4.2 deserves its own evaluation — specifically, how allocation latency holds up as sensor density scales, building on the fog-versus-cloud comparison in Ala'anzy et al. (2025). Adding a demand-forecasting term to the fitness function, using an LSTM or similar model to anticipate near-future arrivals, could let the allocator pre-balance ahead of a predictable rush rather than reacting after the fact. A federated variant would let multiple facilities improve shared demand forecasts without pooling raw occupancy and movement data, which carries its own privacy considerations. And on the algorithm side, hybridising LOA with other bio-inspired operators may speed convergence further for facilities with hundreds of simultaneously free slots, where the batch sizes used here would need to scale up substantially.

REFERENCES

[01] Ala'anzy, M. A., Abilakim, A., Zhanuzak, R., & Li, L. , Real time smart parking system based on IoT and fog computing evaluated through a practical case study. Scientific Reports, 15, Article 33483. https://doi.org/10.1038/s41598-025-15507-6,2025
[02] Al Mamun, A., Hasib, A., Mussa, A. S. M., Hossen, R., & Rahman, A.,IoT-enabled smart car parking system through integrated sensors and mobile applications. arXiv. https://arxiv.org/abs/2412.10774, 2024
[03] M. Aldossary, "A review of dynamic resource management in cloud computing environments," Computer Systems Science and Engineering, vol. 36, no. 3, pp. 461-476, 2021.
[04] C. Chi, Y. Wang, X. Tong, M. Siddula, and Z. Cai, "Game theory in Internet of Things: A survey," IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1-1, 2021.
[05] L. Bhajantri and S. S. Gangadharaiah, "A comprehensive survey on resource management in Internet of Things," Journal of Telecommunications and Information Technology, vol. 4, no. 4, pp. 27-43, 2020.
[06] P. Matta and S. Sharma, "Strategic allocation of resources in an IoT environment: RDSA algorithm," International Journal of Innovative Technology and Exploring Engineering, vol. 9, no. 2, pp. 1110-1114, 2020.
[07] Y. Jie, C. Guo, K.-K. R. Choo, C. Liu, and M. Li, "Game-theoretic resource allocation for fog-based industrial Internet of Things environment," IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1-1, 2020.
[08] M. Bendechache, S. Svorobej, P. Endo, and T. Lynn, "Simulating resource management across the cloud-to-thing continuum: A survey and future directions," Future Internet, vol. 12, no. 6, p. 95, 2020.
[09] F. Kong, Y. Zhou, B. Xia, L. Pan, and L. Zhu, "A security reputation model for IoT health data using S-AlexNet and dynamic game theory in cloud computing environment," IEEE Access, vol. PP, no. 99, pp. 1-1, 2019.
[10] J. Wan, B. Chen, M. Imran, F. Tao, D. Li, C. Liu, and S. Ahmad, "Toward dynamic resources management for IoT-based manufacturing," IEEE Communications Magazine, vol. 56, no. 2, pp. 52-59, 2018.
[11] H. Zhang, Y. Zhang, Y. Gu, D. Niyato, and Z. Han, "A hierarchical game framework for resource management in fog computing," IEEE Communications Magazine, vol. 55, no. 8, pp. 52-57, 2017.
[12] Yazdani, M., & Jolai, F. .Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm. Journal of Computational Design and Engineering, 3(1), 24–36. https://doi.org/10.1016/j.jcde.2015.06.003, 2016.
[13] Nezarat and G. Dastghaibyfard, "Efficient Nash equilibrium resource allocation based on game theory mechanism in cloud computing by using auction," PLOS ONE, vol. 10, no. 10, pp. 1-29, 2015
[14] Rajakumar, B. R,The lion's algorithm: A new nature-inspired search algorithm. Procedia Technology, 6, 126–135. https://doi.org/10.1016/j.protcy.2012.10.016, 2012.
[15] Shoup, D. C., Cruising for parking. Transport Policy, 13(6), 479–486,2006.
[16] Kennedy, J., & Eberhart, R., Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks, 4, 1942–1948, 1995.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)