future internet

MDPI

Article

Resource Assignment Algorithms for Autonomous Mobile Robots with Task Offloading

Giuseppe Baruffa * and Luca Rugini

Department of Engineering, University of Perugia, I-06125 Perugia, Italy
* Correspondence: giuseppe.baruffa@unipg.it

Abstract: This paper deals with the optimization of the operational efficiency of a fleet of mobile robots, assigned with delivery-like missions in complex outdoor scenarios. The robots, due to limited onboard computation resources, need to offload some complex computing tasks to an edge/cloud server, requiring artificial intelligence and high computation loads. The mobile robots also need reliable and efficient radio communication with the network hosting edge/cloud servers. The resource assignment aims at minimizing the total latency and delay caused by the use of radio links and computation nodes. This minimization is a nonlinear integer programming problem, with high complexity. In this paper, we present reduced-complexity algorithms that allow to jointly optimize the available radio and computation resources. The original problem is reformulated and simplified, so that it can be solved by also selfish and greedy algorithms. For comparison purposes, a genetic algorithm (GA) is used as the baseline for the proposed optimization techniques. Simulation results in several scenarios show that the proposed sequential minimization (SM) algorithm achieves an almost optimal solution with significantly reduced complexity with respect to GA.

Keywords: mobile robots; radio resource assignment; task offloading; metaheuristic optimization; latency minimization

Academic Editors: Nguyen Khoa, Steve Drew, Qihao Li and Jinhua Guo

Received: 5 December 2024
Revised: 10 January 2025
Accepted: 14 January 2025
Published: 16 January 2025

Citation: Baruffa, G.; Rugini, L. Resource Assignment Algorithms for Autonomous Mobile Robots with Task Offloading. *Future Internet* **2025**, *17*, 39. https://doi.org/10.3390/fi17010039

1. Introduction

The introduction of autonomous mobile robots in the freight and food delivery chain of smart cities has always been considered a practical application for exploiting the numerous benefits of artificial intelligence (AI) in robotics [1]. Autonomous vehicles (AVs) provide for a transformation of the classical mobility approach on transportation networks and infrastructures by stripping the control and driving supervision from humans and letting AI-enabled vehicles manage themselves with the help of powerful wireless connectivity [2]. Autonomous driving, without human intervention, requires high-performance onboard computation capabilities, advanced sensor and actuator systems, and machine-to-machine communication technologies to carry out geographic and semantic understanding of the city infrastructure, with the consequent sensing, detection, classification, and reasoning tasks [3]. The efficiency of delivery associated with autonomous vehicles implies that their management should be coordinated or centralized so that system costs for the fleet are reduced [4]. Moreover, considering that mobile robots may have limited battery and processing capabilities, their decisions in complex environments can take a long time to be accepted; cloud, edge, and fog computing resources may enhance the quality of these decisions by offering off-board accelerated services to perform complex computation tasks with a lower latency [5–10]. In all these systems, key performance indicators (KPIs) for

efficient offloading in edge computing include latency, data rate, energy consumption, seamless access, and network connectivity, which are typically traded off with throughput, service blockage, handover, and computation accuracy [11].

Figure 1 depicts the general scenario we are considering. A fleet of AVs are connected to a network of servers located in remote data centers (DCs), over which each AV offloads an AI task that, due to its complex nature, is central processing unit (CPU)-intensive. For instance, the AV might request object detection on the scene captured by its onboard cameras: the detection results, coming back from the server, are used to make important decisions for the safety of the AV and/or of persons that are close. In one case, the AV could decide to move along a path that skips particular objects (e.g., obstacle avoidance), while in another case, it could choose to follow a specific object (e.g., person tracking). Note that, depending on the celerity at which responses are received from the servers, these decisions could be more or less delayed, thus negatively impacting the global performance of the task assigned to the AV fleet, which has to fulfill a delivery or exploratory mission by visiting a number of waypoints [12]. This scenario includes obstacles, which have to be recognized and avoided using onboard sensors and (possibly offloaded) object detection services. Image streaming is quite data-intensive, especially if high-resolution or depth cameras are used onboard, so the bit rate produced by the AVs needs high-speed wireless links. For this purpose, a network of base stations (BS) located in the area of interest can be used to access the Internet and, then, to reach the servers. Radio resources are limited, so when multiple AVs try to access the servers using the same BS, the link is overloaded, generating additional delays and an effective access bit rate reduction. Similarly, if multiple AVs offload their tasks to the same DC, its computation resources can be overwhelmed, and the operating system (OS) scheduler or threading model in the service would produce an increased delay in the response. In summary, the typical latency that characterizes each offloading service is incremented by such additional delays, which are encountered either at the radio layer or at the service layer.

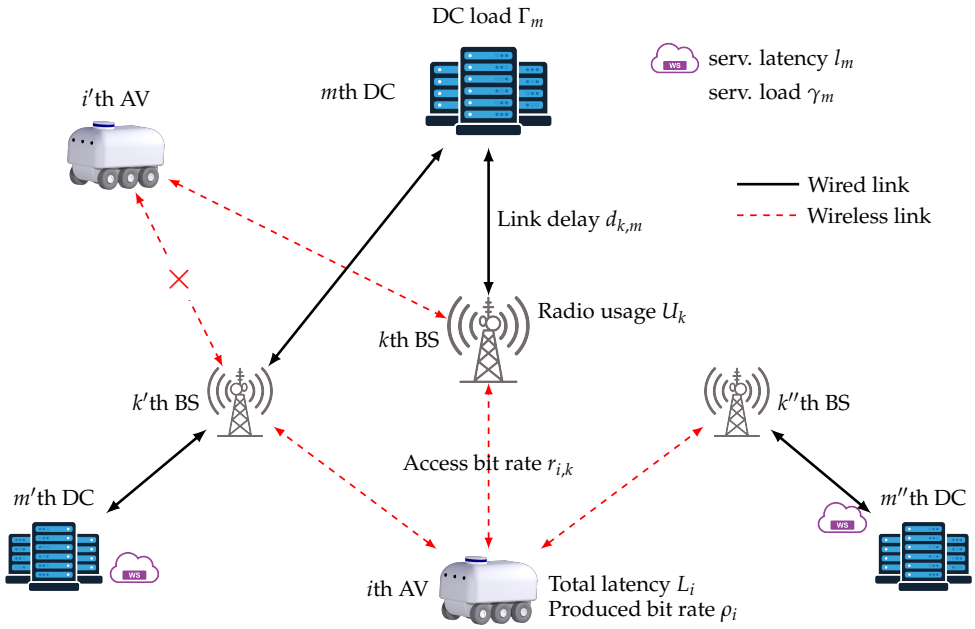


Figure 1. The scenario of interest, comprising a fleet of AVs, a network of BSs, and remote DCs offering AI services.

The described scenario is an evolution of that in [10], which describes a testbed composed of a single AV and up to three different BSs. Namely, the AV is a ground mobile robot equipped with a camera and multiple radio access technologies (Wi-Fi, 5G, etc.),

which moves in a complex indoor environment. Object detection is offloaded to external edge/cloud computers for the purpose of identifying objects in the captured scene, and tracking is employed to follow the object in the environment (person tracking). There, the choice of the access radio UL and of the offloading computer was implemented by a very simple algorithm, which did not take into account a global optimization strategy, which is necessary when multiple AVs struggle to exploit the same radio and computation resources.

In this work, we propose some novel algorithms that can be used to assign the optimal configuration of BSs and DCs that connects each AV to an offloading service so that a selected KPI (e.g., the average latency, the handover rate, the access bit rate) can be optimized. Each AV expects a response from the server after a certain delay or latency, and we chose to minimize the average latency of all AVs. Differently from previous works in the literature, such as [13,14], which use a queuing model, our simpler model accepts all the requests in parallel, and proportionally delays all the responses to satisfy the constraints on the limited radio and computation resources. The concurrent selection of access BS and offloading DC, for all AVs, defines a network configuration that may increase or decrease the average fleet latency, depending on the chosen links and on the possibility of resource overloading. The configuration should be found quickly so that the fleet operation remains functional and optimal also in rapidly changing scenarios. Since this problem falls in the category of combinatorial optimization, we resort to well-known techniques for providing an estimate of the solution, such as random search or metaheuristic algorithms. We also propose a new iterative algorithm, based on sequential minimization, that can produce near-optimal results with a reasonable complexity, with performance comparable to that of exhaustive and metaheuristic methods. The problem of latency minimization has also been investigated in [15], which proposed the minimum latency (MLAT) algorithm. MLAT provides a low-complexity suboptimal solution to the latency minimization; however, we show that, despite the name of the algorithm, the average latency of MLAT is significantly larger than the latency obtained by the sequential minimization algorithm newly proposed in this paper.

This paper is organized as follows: Section 2 discusses previous works on this subject and highlights similarities and differences with our work. The mathematical model of the system introduced here is detailed in Section 3, where we also provide an example, formulate the optimization problem with a mathematical notation, and discuss its complexity. Section 4 is dedicated to the presentation of the proposed algorithms, which try to solve the optimization problem. Simulations are described and their results discussed in Section 5, while conclusions are drawn in Section 6.

2. Literature Review

The problem of jointly managing the radio and computational resources, and finding an optimal assignment for multiple users, has already been coped with in recent works. Thus, for the sake of brevity, only a selection of recent papers is reviewed in the following.

In [16], the authors propose an iterative algorithm to solve the nonconvex problem of optimizing the radio and computation resource assignment among a group of users, while considering bounds on latency and energy consumption. Their algorithm adopts successive convex approximations to achieve the suboptimal solution. They also show that, by leveraging the multiple-input multiple-output (MIMO) capabilities of the BSs, the energy efficiency of the terminals is maximized and the handovers are carefully handled. The same authors also propose another similar method in [17], which is suitable for a distributed implementation in the radio cell controllers. Eriksson et al. [18] decide to assign communication and computation resources to web cameras that provide a multi-view scene to a fog computing system. The quadratic mixed-integer optimization problem is

intractable and they propose an approximation using the Dyer–Zemel algorithm, which is further simplified into an iterative algorithm. A branch-and-bound solution of the mixed-integer nonlinear programming (MINLP) resource allocation problem is presented in [19], which also introduces a solution based on the Gini coefficient, with a predictable polynomial complexity, and may employ a local execution of the task. The authors of [20] consider resource allocation as a game between users and cloud providers, constrained on service delay, transmission quality, and power control. The MINLP problem is solved using a student-project allocation matching game, which may also evolve into user-oriented cooperation to find the optimal solution. Liu and Ansari [21] optimize uplink transmission power, receive beamforming, computation task assignment, and computation resource allocation by reformulating the MINLP into a dual problem solved with the coordinate descent method. The authors of [13] employ a queuing model to describe a complex scenario of thousands of users offloading video-based tasks through Wi-Fi to edge servers. They model also the capacity bottlenecks and present a resource optimization solution using Pareto-optimal edges. In [22], the authors propose EdgeFlow, a mobile edge/cloud computing system, which minimizes the latency by considering two different states and simple convex optimization. Chen et al. [23] transform the mixed-binary nonlinear programming (MBNLP) problem into a geometric programming problem and try to solve this using general bender decomposition. To reduce complexity, this solution is simplified with a recursive algorithm that exploits also some randomness. The authors of [24] show how to simultaneously meet the request of sensing, communication, and computation by employing a carefully crafted wireless scheduling architecture. They rely on multi-attribute decision-making theory to express the problem, reformulated as a two-sided matching problem, and the solution is found in an iterative manner through a modified matching technique. Zhou et al. [14] investigate an ultra-low-latency communication scheme that involves packet request rates, computation latency and rates, communication power, data length, and transmission information amounts. The nondeterministic polynomial (NP)-hard problem is simplified and solved thanks to successive convex approximations, and the efficiency is verified through monotonic optimization. More recently, in [25], the authors propose to execute locally or to offload to neighboring robots some tasks generated by a network of sensors, for jointly minimizing end-to-end delay and energy consumption. They use a deep reinforcement learning approach to plan the best offloading pattern and trajectory of each robot in a restricted scenario, after formulating the problem as a Markov decision process. In [26], instead, the model considers offloading mobile robot tasks on edge servers, and the multi-objective MINLP has to minimize travel times and computation workload. Thanks to the weighted-sum method, Pareto-optimal solutions are found that outperform multiple single-objective approaches based on vehicle routing problem-solving methods. Integrated sensing, computing, and communication are optimized in [27], where a fleet of robots exploits a reflective intelligent surface to extend the radio coverage area in potential blockage conditions, and concurrently senses the environment by means of the communication radio signal. The problem is decoupled by minimizing first the computation latency, and then by maximizing transmission rate and sensing accuracy. Thanks to bisection search and alternating optimization algorithms, significant service quality enhancement, latency reduction, and reliability improvement are obtained with respect to other baseline solutions. Service placement is added as a further variable in [28], extending the classical task offloading, sensing, and communication optimization problem. The authors initially focus on minimizing both the latency and the number of deployed services, and then propose a method to optimize the placement of such services on specific edge/cloud servers. The MINLP problem is solved using a sequential fixing algorithm, with long-term and short-term solution strategies.

A slightly different problem considered in the literature is the assignment of remote radio units (next to antennas) to baseband signal processing units (in remote computers), for the implementation of virtualized radios in cloud radio access networks (CRAN). In this case, the radio access units are an active part of the optimization problem, since they offload some radio data processing tasks to the remote services. In this sense, some of the solutions devised for CRANs fit well with the problem considered in this paper. For instance, in [29], the optimization of CRAN unit assignment is investigated. The authors aim at energy efficiency improvement, and devise a mixed-integer nonconvex programming problem. Then, they reformulate it as a nonlinear fractional problem and solve it with an iterative algorithm based on the subgradient method. Luong et al. [30] rely on a branch-and-bound solution to optimize the system and lower the complexity thanks to a difference of convex algorithm (DCA) method. The authors of [31] use a queuing model for their system and, thanks to auction theory, solve for the optimum with maximum power, radio allocation, interference, and queuing stability constraints. In [32], the integer linear programming formulation of the problem allows finding several algorithms based on matroids, b-matching, and multiple knapsacks. These algorithms optimize the efficiency of network resource utilization while keeping precise latency requirements. To this purpose, the authors employ a cost function that considers both the front network communication latency and the resources used in edge DCs. Shirzad and Gadheri [33] pose the problem as nonconvex and combinatorial, then they propose an approximated solution based on convexification, integer relaxation, and Lagrangian analysis.

In summary, many of the reviewed literature works adopt complex models that account for specific issues of the communication aspect (link budget, blockage probability, radio propagation, etc.), of the computation aspect (requests queue, CPU cycles, energy consumption, etc.), and of the service aspect (neighbor or edge placement, server selection, requests size, etc.). In our model, instead, all such aspects are simplified by considering stationary conditions in short time windows. Moreover, our proposed solutions do not exploit overly complicated algorithms or advanced machine learning techniques, but rather simple greedy procedures following a straightforward minimization approach. We also consider a computationally complex genetic algorithm, which is used to find the MINLP optimal solution for comparison purposes; however, with nonstringent real-time requirements, it could also constitute a viable approach to finding the problem solution.

3. System Model and Problem Formulation

With reference to Figure 1, we introduce the system model that will be used to mathematically formulate the optimization problem in Section 3.2 by adopting a three-level graph structure, such as that shown in Figure 2.

The problem dimensions are given by the number of AVs N_{AV}, the number of BSs N_{BS}, and the number of DCs N_{DC} in the graph levels.

At the leftmost level, the AVs are indexed by i, i', i'', or $\iota = 1, \ldots, N_{\mathrm{AV}}$, they are equipped with an internal service, which is characterized by a latency $\ell_i > 0$, which can perform the assigned task when there is no route configured, and produce an UL bit rate ρ_i.

In the intermediate level, the BSs are indexed by k, k', or $\kappa = 1, \ldots, N_{\mathrm{BS}}$ and provide a UL access bit rate $r_{i,k}$ to an AV, only when the radio connection is active and available, expressed by $D_{i,k} \in \{0, 1\}$ (this corresponds to a presence or absence of the edge connecting the AV node with a BS node). Every AV that is connected to and is using the kth BS contributes to a total radio link usage $U_k \geq 0$, expressing the amount of occupied radio resources.

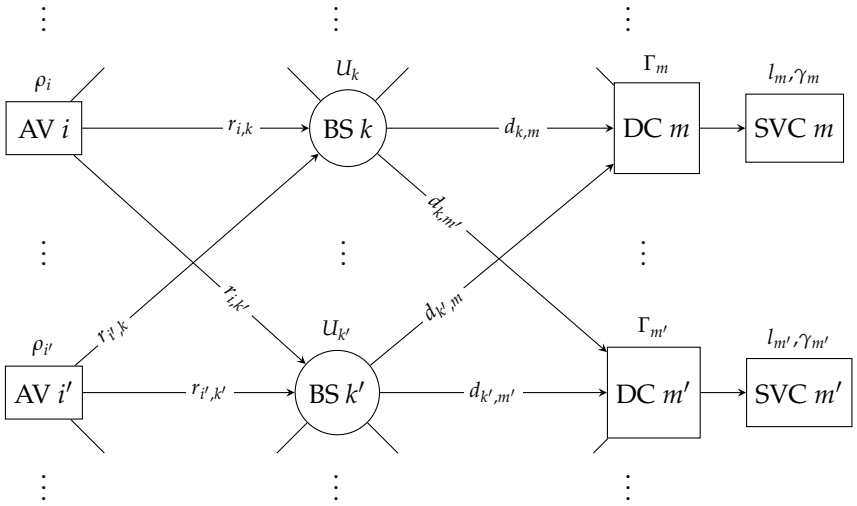


Figure 2. Three-level directed graph modeling the radio connections and the backhaul connections available for assignment. Note that some edges could not be actually present, depending on $D_{i,k}$ and $K_{k,m}$.

In the rightmost level, the DCs are indexed by m, m', or $\mu = 1, \ldots, N_{\mathrm{DC}}$, with $l_m > 0$ denoting the typical end-to-end latency offered by the service and $d_{k,m} > 0$ representing the latency between a BS and a DC; the connection availability between a BS and a DC is expressed by $K_{k,m} \in \{0, 1\}$ (an edge is present or absent in the graph). Note that in Figure 2, we split the rightmost level into two separate sublevels but only for drawing clarity: we consider that each DC hosts a single offloading service (SVC). Every service, when used by an AV, contributes an amount γ_m to the normalized CPU load $\Gamma_m \geq 0$.

The performance generated by a particular assignment of AVs to DCs passing through BSs can be measured by different KPIs, such as the response delay or latency, the average UL access rate, or the number of overloaded DCs, for example.

In order to identify a specific assignment, the vector $\mathbf{f} = \begin{bmatrix} f_1 & \ldots & f_{N_{\mathrm{AV}}} \end{bmatrix}$, $f_i \in \{0, 1\}$ marks the AVs that are using the internal service ($f_i = 1$) rather than the external ones ($f_i = 0$). In addition, the partial selection matrix $\mathbf{S}_i = \begin{bmatrix} S_{i,k,m} \end{bmatrix}$ of size $N_{\mathrm{BS}} \times N_{\mathrm{DC}}$, $S_{i,k,m} \in \{0, 1\}$ chooses the BS-DC pair (k, m) used to offload the task of the ith AV (together with the corresponding $f_i = 0$); there is only a single 1 in the entries so that $\sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} S_{i,k,m} = 1$. The full selection matrix $\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \ldots & \mathbf{S}_{N_{\mathrm{AV}}} \end{bmatrix}$ of size $N_{\mathrm{BS}} \times N_{\mathrm{AV}} N_{\mathrm{DC}}$ represents the complete assignment for all AVs.

Equivalently, the indexing vectors $\mathbf{k} = \begin{bmatrix} k_1 & \ldots & k_{N_{\mathrm{AV}}} \end{bmatrix}$ and $\mathbf{m} = \begin{bmatrix} m_1 & \ldots & m_{N_{\mathrm{AV}}} \end{bmatrix}$ denote the selected pairs of indexes for a specific route, with $(n_i, m_i) = (0, 0)$ flagging the internal service choice. For instance, $k_i = 3$ and $m_i = 2$ means that the ith AV is using the 3rd BS and the 2nd DC. One can commute between the two representations with the mapping function $(\mathbf{f}, \mathbf{S}) = \mathrm{M}(\mathbf{k}, \mathbf{m})$, which converts from the index sets to the selection sets, according to the rules

$$f_i = \delta[k_i m_i], \tag{1}$$

$$S_{i,k,m} = \delta[k - k_i]\delta[m - m_i], \tag{2}$$

where $\delta[\cdot]$ is the Kronecker delta function. Conversely, the demapping function $(\mathbf{k}, \mathbf{m}) = \mathrm{M}^{-1}(\mathbf{f}, \mathbf{S})$, does the opposite according to

$$(k_i, m_i) = \begin{cases} \arg\max_{k,m} S_{i,k,m}, & f_i = 0, \\ (0,0), & f_i = 1. \end{cases} \tag{3}$$

With the help of the selection matrix, we define the radio usage of an AV-BS link as

$$U_k = \sum_{i'=1}^{N_{\mathrm{AV}}} \sum_{m'=1}^{N_{\mathrm{DC}}} \frac{\rho_{i'}}{r_{i',k}} D_{i',k} K_{k,m'} S_{i',k,m'}, \tag{4}$$

and the CPU load of a DC as

$$\Gamma_m = \gamma_m \sum_{i''=1}^{N_{\mathrm{AV}}} \sum_{k'=1}^{N_{\mathrm{BS}}} D_{i'',k'} K_{k',m} S_{i'',k',m}. \tag{5}$$

Let the experienced latency be considered here as the target KPI. Assume that the AV produces N_{req} service requests per time unit; for instance, in the case of video streaming, $N_{\mathrm{req}} = 30$ frames per second should be processed by the remote service. In our model, for every sent video frame, there should be a response from the service, upon which the AV modifies its trajectory or speed. The latency expected by the application managing the AV is thus $l_m = 1/N_{\mathrm{req}}$. Now, we consider the effects of a finite amount of radio and computation resources. The first bottleneck is represented by the radio access UL channel: if multiple AVs are concurrently using the kth BS, which shares its radio resources equally, we should take into account the normalized radio usage U_k. When $U_k \leq 1$, the UL requests (i.e., video frames) pass regularly and without any streaming interruption. However, when $U_k > 1$, the BS controller proportionally reduces the number of requests passing through so that it effectively forwards only $N_{\mathrm{FW}} = \frac{N_{\mathrm{req}}}{U_k}$ requests per time unit toward the DC [13]. Similarly, the DCs represent another bottleneck. If too many AVs are using the same DC, its CPU can get overloaded so that $\Gamma_m > 1$. Then, if the OS scheduler is fair, it will proportionally assign processes a lower CPU time to perform computations [34], thus decreasing the effective number of requests served to $N_{\mathrm{SRV}} = \frac{N_{\mathrm{FW}}}{\Gamma_m}$. For the AV, it is as if the effective latency experienced by the application becomes $L_i = \frac{1}{N_{\mathrm{SRV}}} = l_m U_k \Gamma_m$. Thus, considering (4)–(5) and the network delay $d_{k,m}$ on the wired BS-DC connections, we can write the latency experienced by the ith AV as

$$L_i = \begin{cases} l_{m_i} \max\{1, U_{k_i}\} \max\{1, \Gamma_{m_i}\} + d_{k_i,m_i}, & f_i = 0 \wedge \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} D_{i,k} K_{k,m} S_{i,k,m} = 1, \\ \infty, & f_i = 0 \wedge \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} D_{i,k} K_{k,m} S_{i,k,m} = 0, \\ \ell_i, & f_i = 1. \end{cases} \tag{6}$$

The value $L_i = \infty$ in (6) represents the case when the selection does not entail a valid path between the AV and the BS or between the BS and the DC.

3.1. Example

An example of configuration is shown in the graph of Figure 3, which shows all the possible connections and, highlighted, a specific resource assignment. The dimensions of this example are $N_{\mathrm{AV}} = 4$, $N_{\mathrm{BS}} = 3$, and $N_{\mathrm{DC}} = 2$, while the latencies $d_{k,m}$, l_m, and ℓ_i, the loads γ_m, the bit rates $r_{i,k}$ and ρ_i, and the connection matrixes $D_{i,k}$ and $K_{k,m}$ are printed in Table 1 (a null value in $d_{k,m}$ corresponds to a missing link between the BS and the AV).

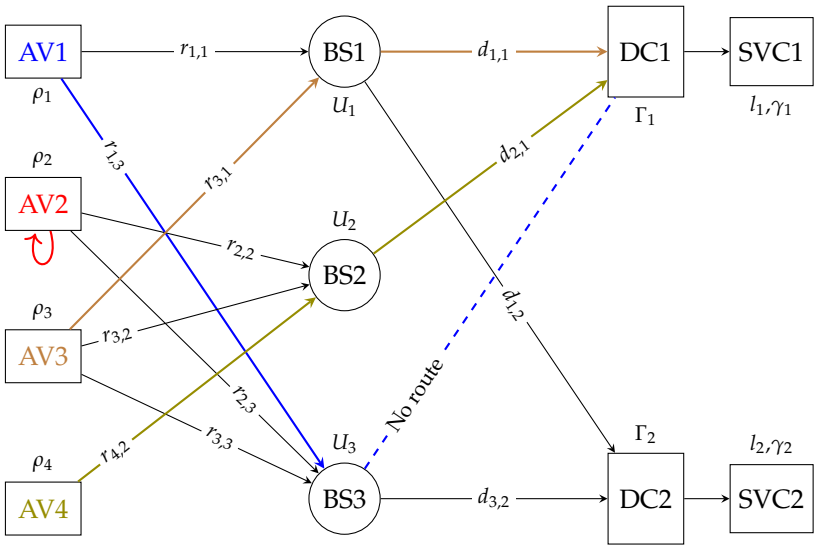


Figure 3. Graph of assignment in Table 2 for the example in Table 1.

Table 1. Input parameters of the example in Figure 3.

Parameter	Value
l_m (ms)	$[33 \quad 33]$
ℓ_i (ms)	$[100 \quad 100 \quad 100 \quad 100]$
γ_m	$[1/4 \quad 1/4]$
ρ_i (Mbit/s)	$[5 \quad 5 \quad 8 \quad 5]$
$r_{i,k}$ (Mbit/s)	$\begin{bmatrix} 14 & 0 & 2 \\ 0 & 12 & 8 \\ 7 & 9 & 8 \\ 0 & 8 & 0 \end{bmatrix}$
$D_{i,k}$	$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
$d_{k,m}$ (ms)	$\begin{bmatrix} 24 & 6 \\ 18 & 0 \\ 0 & 21 \end{bmatrix}$
$K_{k,m}$	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

A possible assignment $(\mathbf{f}, \mathbf{S})$ is specified in Table 2, and it has been highlighted by using different colors and thicker lines in Figure 3.

Table 2. A possible assignment for the example in Figure 3.

Parameter	Value
$\mathbf{f}$	$[0 \quad 1 \quad 0 \quad 0]$
$\mathbf{S}_1$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$
$\mathbf{S}_2$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$

Table 2. *Cont.*

Parameter	Value
$\mathbf{S}_3$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$
$\mathbf{S}_4$	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$

The assignment results in the following radio usage ratios

$$U_1 = \frac{\rho_3}{r_{3,1}} = \frac{8}{7} \approx 1.14, \tag{7}$$

$$U_2 = \frac{\rho_4}{r_{4,2}} = \frac{5}{8} = 0.625, \tag{8}$$

$$U_3 = 0, \tag{9}$$

which shows that the radio link deployed at BS1 is overexploited. Similarly, the CPU loads are calculated as

$$\Gamma_1 = 2\gamma_1 = 0.5, \tag{10}$$

$$\Gamma_2 = 0, \tag{11}$$

so the two DCs are not overloaded. In terms of experienced latency, AV1 does not have a route toward any service, so it is $L_1 = \infty$. AV2, instead, uses its internal service, experiencing a latency $L_2 = \ell_2 = 100\,\text{ms}$. The third AV offloads its task to SVC1 on DC1, passing through BS1, which is overloaded at 114% of the link capacity; the typical latency l_1 is thus multiplied by $U_1 \approx 1.14$ and increased by the wired connection latency $d_{1,1} = 24\,\text{ms}$, achieving $L_3 = l_1 U_1 + d_{1,1} \approx 61.62\,\text{ms}$. Finally, AV4 also has a link to the service SVC1 that passes through a nonoverloaded BS; the experienced latency is $L_4 = l_1 + d_{2,1} \approx 51\,\text{ms}$.

3.2. Formulation of the Minimization Problem

The problem consists in finding a specific configuration $(\mathbf{f}, \mathbf{S})$ that optimizes a KPI $\mathcal{P}$, such as, for instance, the latency, bit rate, or radio/CPU loads. In this work, we want to minimize the average latency experienced by all AVs, written as

$$\mathcal{P}(\mathbf{f}, \mathbf{S}) = \frac{1}{N_{\text{AV}}} \sum_{i=1}^{N_{\text{AV}}} L_i(\mathbf{f}, \mathbf{S}), \tag{12}$$

while choosing a unique configuration that connects the AV to a DC through a BS and a finite amount of radio and computation resources. Thus, as described in Section 3 and discussed in the example, every connection to a DC is unique, i.e., it cannot be split among two or more BSs. Similarly, every AV can connect to a single DC only, but differently, it must use its internal service. Due to finite radio and computation resources, latency increases proportionally when either BSs or DCs are overloaded, as shown by (6). With these conditions, the problem falls within the case of nonlinear integer (binary) constrained optimization, and, since the number of AVs is fixed, can be mathematically formulated as a minimization of the total sum latency, as expressed by

$$\min_{\mathbf{f},\mathbf{S}} \quad \sum_{i=1}^{N_{\mathrm{AV}}} L_i(\mathbf{f},\mathbf{S}) \tag{13}$$

$$\text{subject to} \quad \mathbf{f} \in \{0,1\}^{N_{\mathrm{AV}}} \tag{14}$$

$$\mathbf{S} \in \{0,1\}^{N_{\mathrm{BS}} \times N_{\mathrm{DC}} N_{\mathrm{AV}}} \tag{15}$$

$$f_i + \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} S_{i,k,m} = 1, \; i = 1, \ldots, N_{\mathrm{AV}}. \tag{16}$$

We rewrite L_i in (6) as

$$\begin{aligned} L_i(\mathbf{f},\mathbf{S}) =& \ell_i f_i \\ &+ \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} [(l_m \max\{1, U_k\} \max\{1, \Gamma_m\} + d_{k,m}) D_{i,k} K_{k,m} + l_\infty (1 - D_{i,k} K_{k,m})] S_{i,k,m}, \\ & i = 1, \ldots, N_{\mathrm{AV}}, \end{aligned} \tag{17}$$

so as to make explicit its dependency on the assigned configuration $(\mathbf{f},\mathbf{S})$ and to contain the various cases into a single equation. To include the 2nd line in the right-hand side of (6), we also introduce $l_\infty \gg l_m$, a latency value denoting connection absence (e.g., $l_\infty = 10^9$). Expression (17), then, represents the latency distribution among the AVs. The problem (13) is integer nonlinear, and also nonconvex, where the constraints are as follows:

- (14) and (15) state that the selection vector and matrix have binary components;
- (16) states that when $f_i = 1$, then $\mathbf{S}_i = \mathbf{0}$, and that when $f_i = 0$, then there is only a 1 in $\mathbf{S}_i$ (the other values being 0).

In the following, when referring to the complexity of the proposed algorithms, we denote with "*call*" the calculation of the values $L_i(\mathbf{f},\mathbf{S})$ in (17) for $i = 1, \ldots, N_{\mathrm{AV}}$.

4. Proposed Optimization Algorithms

The number of possible configurations allowed by a fully connected graph is

$$N_{\mathrm{all}} = (N_{\mathrm{BS}} N_{\mathrm{DC}} + 1)^{N_{\mathrm{AV}}}, \tag{18}$$

while the number of valid configurations (for which there is a physical link from the AV to the DC) is

$$N_{\mathrm{val}} = \prod_{i=1}^{N_{\mathrm{AV}}} \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} D_{i,k} K_{k,m}. \tag{19}$$

The number N_{val} of valid configurations in (19) can be large, and is bounded by N_{all} in (18), which is exponential in the number of AVs. Therefore, an exhaustive search can be intractable. Hence, in the following, to solve the minimization (3.2), we reformulate the problem to obtain solutions with tractable complexity. Algorithm complexity can be deduced from the number of calls to the latency distribution Formula (17), which has complexity order $\mathcal{O}(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}})$.

4.1. Brute Force Approach: VEX Algorithm

The brute force solution consists in exhaustively enumerating all possible valid configurations, and finding the one that ensures the minimum value of the latency (17); we denote this algorithm as a valid exhaustive (VEX) search. The complexity order of the VEX is $\mathcal{O}\big(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}} (\tilde{N}_{\mathrm{BS}} \tilde{N}_{\mathrm{DC}})^{N_{\mathrm{AV}}}\big)$, where $\tilde{N}_{\mathrm{BS}} = \max_i \sum_{k=1}^{N_{\mathrm{BS}}} D_{i,k} \leq N_{\mathrm{BS}}$ and $\tilde{N}_{\mathrm{DC}} = \max_k \sum_{m=1}^{N_{\mathrm{DC}}} K_{k,m} \leq N_{\mathrm{DC}}$.

4.2. Random Search: RSAM Algorithm

If the number of valid configurations is too high to be computed in a reasonable amount of time, we can use some form of simple stochastic optimization, where we randomly search the valid configurations space [35] to obtain the random sampling (RSAM) algorithm. The complexity order of RSAM depends on the number of calls, as $\mathcal{O}(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}} N_{\mathrm{call}})$.

4.3. Metaheuristic Optimization: GA Algorithm

Many metaheuristic algorithms are nature-inspired and have proven useful in solving a vast set of optimization problems [36]. We use an evolutionary strategy like the genetic algorithm (GA) [37]. In this algorithm, a population of potential solutions (chromosomes) is let to evolve among successive generations. Each generation passes its genes to an offspring thanks to operators such as crossover and mutation. The chromosome of the eth individual in the GA population at the qth generation is specified as

$$\mathbf{g}_e^{(q)} = [g_{i,e}^{(q)}] = \left[k_{1,e} \quad m_{1,e} \quad k_{2,e} \quad m_{2,e} \quad \dots \quad k_{N_{\mathrm{AV}},e} \quad m_{N_{\mathrm{AV}},e}\right], \tag{20}$$

and its fitness is directly calculated as $\varphi_e^{(q)} = \mathcal{P}(\mathrm{M}(\mathbf{g}_e^{(q)}))$. The initial population of N_{GA} individuals is generated with genes randomly distributed among the admissible values, i.e., $g_{i,e}^{(0)} \sim \mathcal{U}[0, g_{i_{\mathrm{MAX}}}]$, with $g_{i_{\mathrm{MAX}}}$ being N_{BS} for the elements in the odd positions in (20) and N_{DC} for the elements in the even positions. Another possibility is to assign to the initial population the internal service configuration, i.e., $g_{i,e}^{(0)} = 0$. To pass their genes among generations, parent chromosomes are chosen with fitness-proportionate selection (roulette wheel). The crossover operation generates offsprings by uniformly exchanging the parents' alleles, $(\mathbf{g}_{e'}^{(q+1)}, \mathbf{g}_{e''}^{(q+1)}) = \mathrm{XOV}(\mathbf{g}_{e'}^{(q)}, \mathbf{g}_{e''}^{(q)})$: a pair (g_{2i}, g_{2i+1}) is exchanged in the offsprings with probability $p_{\mathrm{X}} = 0.5$. In addition, 80% of the population is chosen for crossover, while the remaining 20% of the population does not withstand crossover. A graphical example of the crossover operation is shown in Figure 4.

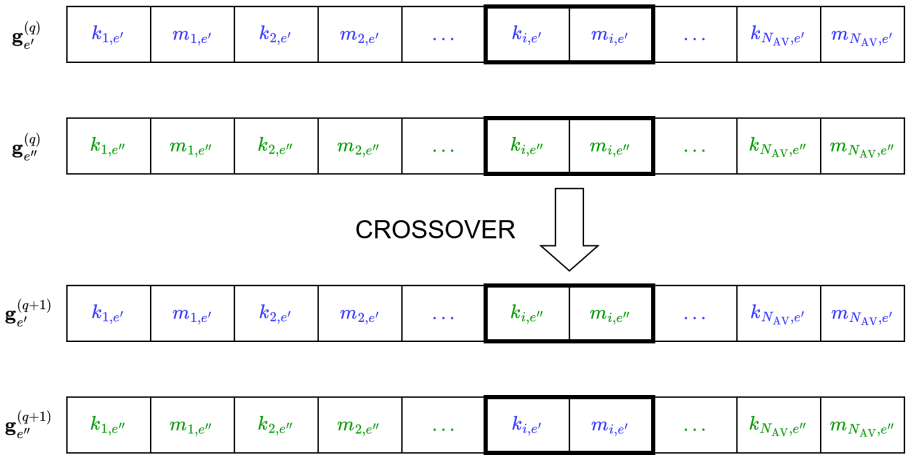


Figure 4. Graphical overview of the crossover operation in the GA algorithm.

The mutation operation, similarly to crossover, randomly changes a pair (g_{2i}, g_{2i+1}) with probability $p_{\mathrm{M}} = 0.01$ to another admissible value, $\mathbf{g}_{e'}^{(q+1)} = \mathrm{MUT}(\mathbf{g}_{e'}^{(q)})$; a pair (g_{2i}, g_{2i+1}) assumes random values chosen among the possible valid configurations for AV i, as shown in Figure 5.

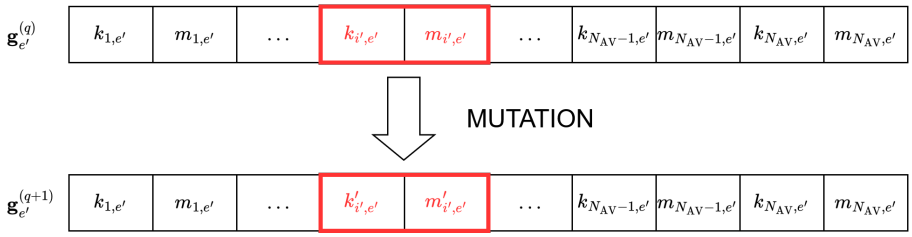


Figure 5. Graphical overview of the mutation operation in the GA algorithm.

An elite subpopulation (top 5% of individuals with the highest fitness) remains unchanged among the generations. The $\hat{e}$th chromosome with the highest fitness,

$$\hat{\varphi}^{(q)} = \varphi_{\hat{e}}^{(q)}, \quad \hat{e} = \arg\max_{e} \varphi_{e}^{(q)}, \tag{21}$$

represents a candidate solution. The GA search process stops either when the best fitness value stalls for q_{STL} generations,

$$\hat{\varphi}^{(q)} = \hat{\varphi}^{(q-1)} = \ldots = \hat{\varphi}^{(q-q_{\mathrm{STL}})}, \tag{22}$$

or when a maximum number q_{MAX} of generations has occurred. The complexity depends on the convergence speed of the GA population to the exact solution and on the number of calls N_{call} to the latency expression (17), so it is $\mathcal{O}(N_{\mathrm{AV}}N_{\mathrm{BS}}N_{\mathrm{DC}}N_{\mathrm{call}})$. (We do not consider the complexity of selection, crossover, and mutation. Instead, in Section 5, the cost of these operations is counted in the total running time.)

4.4. Selfish Reformulation: MLAT Algorithm

Upon neglecting the finite-resource constraints due to (4) and (5), the max{} operators in (17) disappear, and the obtained latency may be rewritten as

$$\begin{aligned} \tilde{L}_i(f_i, \mathbf{S}_i) =& \ell_i f_i + \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} [(l_m + d_{k,m}) D_{i,k} K_{k,m} + l_\infty (1 - D_{i,k} K_{k,m})] S_{i,k,m}, \\ & i = 1, \ldots, N_{\mathrm{AV}}. \end{aligned} \tag{23}$$

Additionally, we reformulate (3.2) as a multi-objective problem,

$$\begin{aligned} \min_{\mathbf{f},\mathbf{S}} \quad & (\tilde{L}_1(f_1, \mathbf{S}_1), \ldots, \tilde{L}_{N_{\mathrm{AV}}}(f_{N_{\mathrm{AV}}}, \mathbf{S}_{N_{\mathrm{AV}}})) \\ \text{subject to} \quad & \mathbf{f} \in \{0,1\}^{N_{\mathrm{AV}}} \\ & \mathbf{S} \in \{0,1\}^{N_{\mathrm{BS}} \times N_{\mathrm{DC}} N_{\mathrm{AV}}} \\ & f_i + \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} S_{i,k,m} = 1, \; i = 1, \ldots, N_{\mathrm{AV}}, \end{aligned} \tag{24}$$

which can be viewed as a selfish (noncooperative) reformulation by letting each AV choose its own minimum-latency assignment. Separability of the objective function and of the constraints $\min L_i(f_i, \mathbf{S}_i)$ leads to a simplified solution: since $l_m + d_{k,m} \ll l_\infty$ in (23), the minimization requires $D_{i,k}K_{k,m} = 1$, yielding the solution

$$f_i^* = \begin{cases} 0, & \min_{k,m:D_{i,k}K_{k,m}=1}(l_m + d_{k,m}) < \ell_i, \\ 1, & \text{else}, \end{cases} \tag{25}$$

$$\mathbf{S}_i^* = \begin{cases} \mathbf{E}_{k_i^*,m_i^*}, & \min_{k,m:D_{i,k}K_{k,m}=1}(l_m + d_{k,m}) < \ell_i, \\ \mathbf{0}_{N_{\mathrm{BS}} \times N_{\mathrm{DC}}}, & \text{else}, \end{cases} \tag{26}$$

where $\mathbf{E}_{k',m'}$ is a matrix whose element in position (k', m') is equal to 1 and zero elsewhere, and

$$(k_i^*, m_i^*) = \begin{cases} \arg\min_{k,m:D_{i,k}K_{k,m}=1}(l_m + d_{k,m}), & \min(l_m + d_{k,m}) < \ell_i, \\ (0,0), & \text{else}, \end{cases} \tag{27}$$

$i = 1, \ldots, N_{\mathrm{AV}}$, $k = 1, \ldots, N_{\mathrm{BS}}$, and $m = 1, \ldots, N_{\mathrm{DC}}$. Note that this solution does not consider overloading of the resources, because the minimization is performed over $\tilde{L}_i(f_i, \mathbf{S}_i)$ in (23); therefore, the true AV latency $L_i(\mathbf{f}^*, \mathbf{S}^*)$ in (17) may be higher than $\tilde{L}_i(f_i^*, \mathbf{S}_i^*)$. The algorithm complexity order is $\mathcal{O}(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}})$.

4.5. Greedy Approximation: SM Algorithm

We approximate the problem (3.2) by using a greedy procedure, where each AV optimizes its own configuration without modifying that of the others. We use the natural ordering of the AVs. Since the minimizations are performed successively, we define this approximation as a sequential minimization (SM). We reformulate the original problem by considering an iterative minimization procedure to find partial solutions. By defining $q = 0, 1, 2, \ldots$ as the iteration index, $(\mathbf{f}^{(q)}, \mathbf{S}^{(q)})$ as the optimal configuration found at the qth iteration, and $f_i^{(0)} = 1$ with $S_{i,k,m}^{(0)} = 0$ as the initial configuration of the ith AV, we rewrite the problem (3.2) as

$$\begin{aligned} \min_{f_{i'}^{(q)}, \mathbf{S}_{i'}^{(q)}} \quad & \sum_{i=1}^{N_{\mathrm{AV}}} L_i(\mathbf{f}^{(q)}, \mathbf{S}^{(q)}) \\ \text{subject to} \quad & \mathbf{f}^{(q)} \in \{0,1\}^{N_{\mathrm{AV}}} \\ & \mathbf{S}^{(q)} \in \{0,1\}^{N_{\mathrm{BS}} \times N_{\mathrm{DC}} N_{\mathrm{AV}}} \\ & f_i^{(q)} + \sum_{k=1}^{N_{\mathrm{BS}}} \sum_{m=1}^{N_{\mathrm{DC}}} S_{i,k,m}^{(q)} = 1, \; i = 1, \ldots, N_{\mathrm{AV}}, \\ & f_i^{(q)} = f_i^{(q-1)}, \forall i \neq i', \; i = 1, \ldots, N_{\mathrm{AV}}, \\ & \mathbf{S}_i^{(q)} = \mathbf{S}_i^{(q-1)}, \forall i \neq i', \; i = 1, \ldots, N_{\mathrm{AV}}, \end{aligned} \tag{28}$$

where i' is the index of the AV to be optimized in the qth iteration, defined as

$$i' = \begin{cases} 1 + (q-1) \bmod N_{\mathrm{AV}}, & q \leq \vartheta N_{\mathrm{AV}}, \\ \arg\max_i L_i(\mathbf{f}^{(q-1)}, \mathbf{S}^{(q-1)}), & q > \vartheta N_{\mathrm{AV}}, \end{cases} \tag{29}$$

$q > 0$, and $\vartheta \geq 1$ defines the number of times all the AVs are sequentially processed. The process stops after q_{STOP} iterations such that

$$L_i(\mathbf{f}^{(q_{\mathrm{STOP}})}, \mathbf{S}^{(q_{\mathrm{STOP}})}) = L_i(\mathbf{f}^{(q_{\mathrm{STOP}}-1)}, \mathbf{S}^{(q_{\mathrm{STOP}}-1)}) \wedge q_{\mathrm{STOP}} > \vartheta N_{\mathrm{AV}} \, . \tag{30}$$

Note that this minimization procedure is greedy: at each step, each AV tries to minimize the average latency of all AVs, exploring only its own assignment subset, conditioned on

the assignment at the previous iteration for the other AVs. The resulting procedure is listed in Algorithm 1.

Algorithm 1 Sequential minimization (SM) of average latency.

Require: $\vartheta \geq 1$
  $(\mathbf{k}^*, \mathbf{m}^*) \leftarrow (\mathbf{0}_{1\times N_{\mathrm{AV}}}, \mathbf{0}_{1\times N_{\mathrm{AV}}})$ ▷ Solution holder
  $(\mathbf{k}, \mathbf{m}) \leftarrow (\mathbf{0}_{1\times N_{\mathrm{AV}}}, \mathbf{0}_{1\times N_{\mathrm{AV}}})$ ▷ Initial and current configuration
  $P^* \leftarrow \infty$ ▷ Solution KPI
  $q \leftarrow 0$ ▷ Flag variable
  while true **do** ▷ Infinite loop
    $q \leftarrow q + 1$ ▷ Increment flag
    if $q \leq \vartheta N_{\mathrm{AV}}$ **then** ▷ Optimize AVs one by one
      $i \leftarrow 1 + (q-1) \bmod N_{\mathrm{AV}}$ ▷ Enumerate all AVs
    else ▷ Optimize the slowest
      $i = \arg\max_{i'} L_{i'}(\mathrm{M}(\mathbf{k}, \mathbf{m}))$ ▷ Find the highest latency
    end if
    $(\mathbf{k}, \mathbf{m}) \leftarrow \arg\min_{k_i, m_i} \mathcal{P}(\mathrm{M}(\mathbf{k}, \mathbf{m}))$ ▷ Improve KPI on a single AV
    if $\mathcal{P}(\mathrm{M}(\mathbf{k}, \mathbf{m})) < P^*$ **then** ▷ There is some improvement in the KPI
      $(\mathbf{k}^*, \mathbf{m}^*) \leftarrow (\mathbf{k}, \mathbf{m})$ ▷ Better configuration found
      $P^* \leftarrow \mathcal{P}(\mathrm{M}(\mathbf{k}^*, \mathbf{m}^*))$ ▷ Better KPI found
    else ▷ There is no KPI improvement
      if $q > \vartheta N_{\mathrm{AV}}$ **then** ▷ Did the first phase
        break ▷ Exit from infinite loop
      end if
    end if
  end while

During an initial phase, the algorithm explores the solution subspace where every AV is optimized sequentially, one by one, while keeping the other AVs' configurations unchanged. This means that for the specific AV, a configuration is found that produces the lowest average latency among all AVs by enumerating all the valid configurations for that AV. This sequential optimization is repeated ϑ times for every AV.

Then, in the final phase, the worst AV is chosen (the one with the highest latency), and its configuration is potentially improved with an enumeration search, as in the initial phase. This step is repeated (potentially with a different AV) as long as there is some improvement in the average latency of all AVs. The final configuration is considered to be the solution to the problem.

Since in each iteration the number of explored configurations is at maximum $\tilde{N}_{\mathrm{BS}}\tilde{N}_{\mathrm{DC}} + 1$ only, the solution of (28) can be found by exhaustive search. Considering a number of iterations $\vartheta N_{\mathrm{AV}}$, the latency Formula (17) is called $\vartheta N_{\mathrm{AV}}(\tilde{N}_{\mathrm{BS}}\tilde{N}_{\mathrm{DC}} + 1)$ times. The computation of (17) has complexity $\mathcal{O}(N_{\mathrm{AV}}N_{\mathrm{BS}}N_{\mathrm{DC}})$; therefore, the complexity order of SM is $\mathcal{O}\left(\vartheta N_{\mathrm{AV}}^2 N_{\mathrm{BS}}N_{\mathrm{DC}}\tilde{N}_{\mathrm{BS}}\tilde{N}_{\mathrm{DC}}\right)$.

5. Simulation Results and Discussion

We tested the algorithms of Section 4 in different scenarios, for which we consider high-speed ULs implemented with IEEE 802.11ad [38] and 5G NR [39]. This scenario is compatible with the testbed hardware (HW) in [10], where a mobile robot uses both IEEE 802.11ad and 5G connections to offload an object detection AI task. In that case, a tracking application needs, hypothetically, to detect objects from a video stream at about 30 frames per second to operate with good performance. This corresponds to a typical service latency of 33 ms if the service is placed in powerful edge or cloud computers. This latency may exceed 100 ms if the service runs in the AV CPU, which is less powerful. The network delay experienced when connecting to an edge server using the IEEE 802.11ad link is of

few ms, while it rises to dozens of ms if the service lies in the cloud and is accessed through a 5G link. The access bit rates, correspondingly, may be as large as about 100 Mbit/s for 5G or about 2 Gbit/s for IEEE 802.11ad. More in general, the problem dimensions might correspond to systems composed of few elements, such as in an industrial environment, or to larger systems composed of several AVs and dozens of BSs and DCs, such as in a smart city. We present the performance of our algorithms with a few typical scenarios (a detailed description is given in Sections 5.1–5.3), which can be classified as follows:

- *6/7/4_fixed*, *6/7/4_variable*: These correspond to a small–medium industrial application managing intelligent vehicles used for internal logistics [40]. For instance, a small fleet of parts delivery robots moves both inside and about the enterprise location, which is covered by cheap but fast access points and served by a powerful remote cloud data center and a few in-premises edge computers. The robots could make use of computer vision to match the moved goods and verify their integrity status.
- *10/25/13_fixed*, *10/25/5_variable*: These correspond to a medium–large deployment of mobile robots in a medium city to be used for services such as food/meal delivery [41]. In this case, the city wireless network can be used to offer computer vision and lidar-related services to the robots or drones to extend their operational range by offloading such tasks instead of consuming their internal energy.
- *40/60/20_variable*: This scenario corresponds to a large-sized fleet of mobile robots, which could be used for example as autonomous street sweepers in a big smart city [42]. The complex scenario in which they move, characterized by varying obstacles and constraints, requires centralized management, and task offloading accelerates video-based litter detection, lidar-based map augmentation, and cooperative data fusion to optimize the paths and times.

For the simulated scenarios, we present some of the resulting KPIs, notably the total and average latency of the configuration, the average radio load, the number V_{BS} of radio-overloaded BSs, the average DC load, the number V_{DC} of CPU-overloaded DCs, the number N_{call} of calls to (17), and the running time. (Simulations are performed with MATLAB R2024a on an AMD Ryzen 9 4900H CPU running at 3.3 GHz.) Additionally, we show the solution on the connection graph and report a summary of the optimal configurations.

In the fixed case, the parameters (connection matrices, bit rates, delays) are kept fixed for the whole simulation. In the variable case, new randomly generated parameters are used within the simulation, and the averaged results are shown.

As for the comparison between the algorithms, where feasible, since VEX achieves the real optimum value, it is considered as a reference. When the problem is intractable, we consider GA to approximate the real solution, and it will serve as a reference.

5.1. 6/7/4_Fixed Scenario

Scenario *6/7/4_fixed* is simple, as it is characterized by $N_{\mathrm{AV}} = 6$, $N_{\mathrm{BS}} = 7$, and $N_{\mathrm{DC}} = 4$. The service in each DC has typical latency $l_m = 33$ ms and typical load $\gamma_m = 1/4$ CPU, $m = 1, 2, 4$, and $\gamma_3 = 1/8$ CPU. The internal service latency is $\ell_i = 111$ ms, while the bit rates produced by AVs are 32 Mbit/s, 22 Mbit/s, 35 Mbit/s, 24 Mbit/s, 31 Mbit/s and 17 Mbit/s. The connection matrix is seen in the graph of Figure 6, while access bit rates and latency values are shown on the AV-BS and BS-DC edges, respectively. In this scenario, BS7 is a 5G station connected to a cloud server (DC3) with a slow link, while all other BS are IEEE 802.11ad connected to edge servers with fast links. The number of possible configurations is $N_{\mathrm{all}} = 5.9 \times 10^8$, of which $N_{\mathrm{val}} = 3.6 \times 10^3$ are valid.

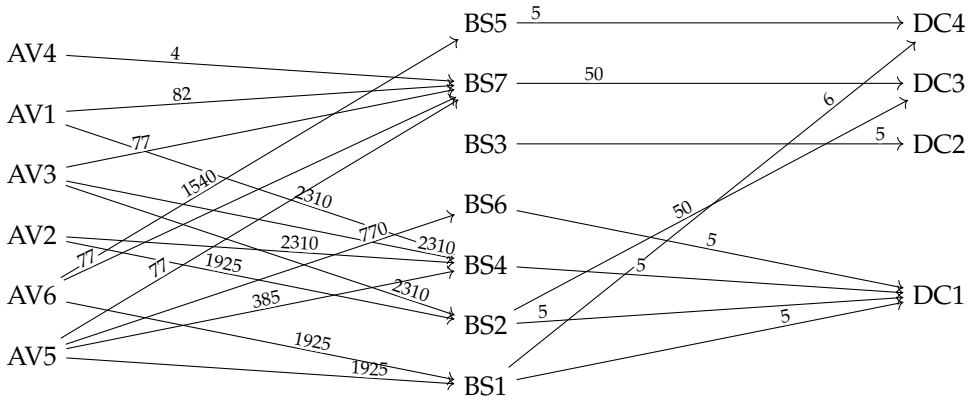


Figure 6. Connection graph of the *6/7/4_fixed* scenario.

For the RSAM algorithm, we chose $N_{\text{call}} = 5 \times 10^4 \gg N_{\text{val}}$; thus, it has a high probability of finding the optimal solution. Indeed, RSAM may generate a configuration that is already generated; hence, $N_{\text{call}} = N_{\text{val}}$ does not guarantee the optimal solution. VEX is simulated since N_{val} is a tractable number of configurations, while for GA, we have set a maximum of $N_{\text{call}} = 2 \times 10^6$ calls. Finally, SM has $\vartheta = 2$.

The KPIs for the found solutions are reported in Table 3. For all the KPIs, all algorithms except MLAT attain the minimum value: 301.0 ms of total latency, 50.2 ms of average latency, 1% of average radio load without any radio overload, 31% of average CPU load, and no DC overloads. As for the computation cost, GA is the most complex, with about 10^6 actual function calls and a 17.8 s run time, followed by RSAM with a 0.4 s run time. Then, VEX and SM have very low complexity, with 3600 and 65 calls, respectively, which are performed in about 15 ms. Finally, MLAT is the fastest, with just 7 ms of run time (note that MLAT does not make any call, but its complexity is roughly equivalent to that of 1 call).

Table 3. KPIs for the solutions to the *6/7/4_fixed* scenario.

	RSAM	**VEX**	**MLAT**	**GA**	**SM**
$\sum_i L_i$ (ms)	301.0	301.0	438.0	301.0	301.0
$\frac{1}{N_{\text{AV}}} \sum_i L_i$ (ms)	50.2	50.2	73.0	50.2	50.2
$\frac{1}{N_{\text{BS}}} \sum_k U_k$	0.01	0.01	0.87	0.01	0.01
V_{BS}	0	0	1	0	0
$\frac{1}{N_{\text{DC}}} \sum_m \Gamma_m$	0.31	0.31	0.34	0.31	0.31
V_{DC}	0	0	0	0	0
N_{call}	50,000	3600	1	953,440	65
Run time (s)	0.418	0.013	0.007	17.830	0.015

The configurations obtained by the simulated algorithms produce the connection graphs shown in Figure 7a–e. The optimal configurations indicate that AV4 has been assigned to the internal service and that, generally, BS4 is preferred for the radio connection and DC1 for the task offloading. Note that in this case VEX and SM produce the same (optimal) solution.

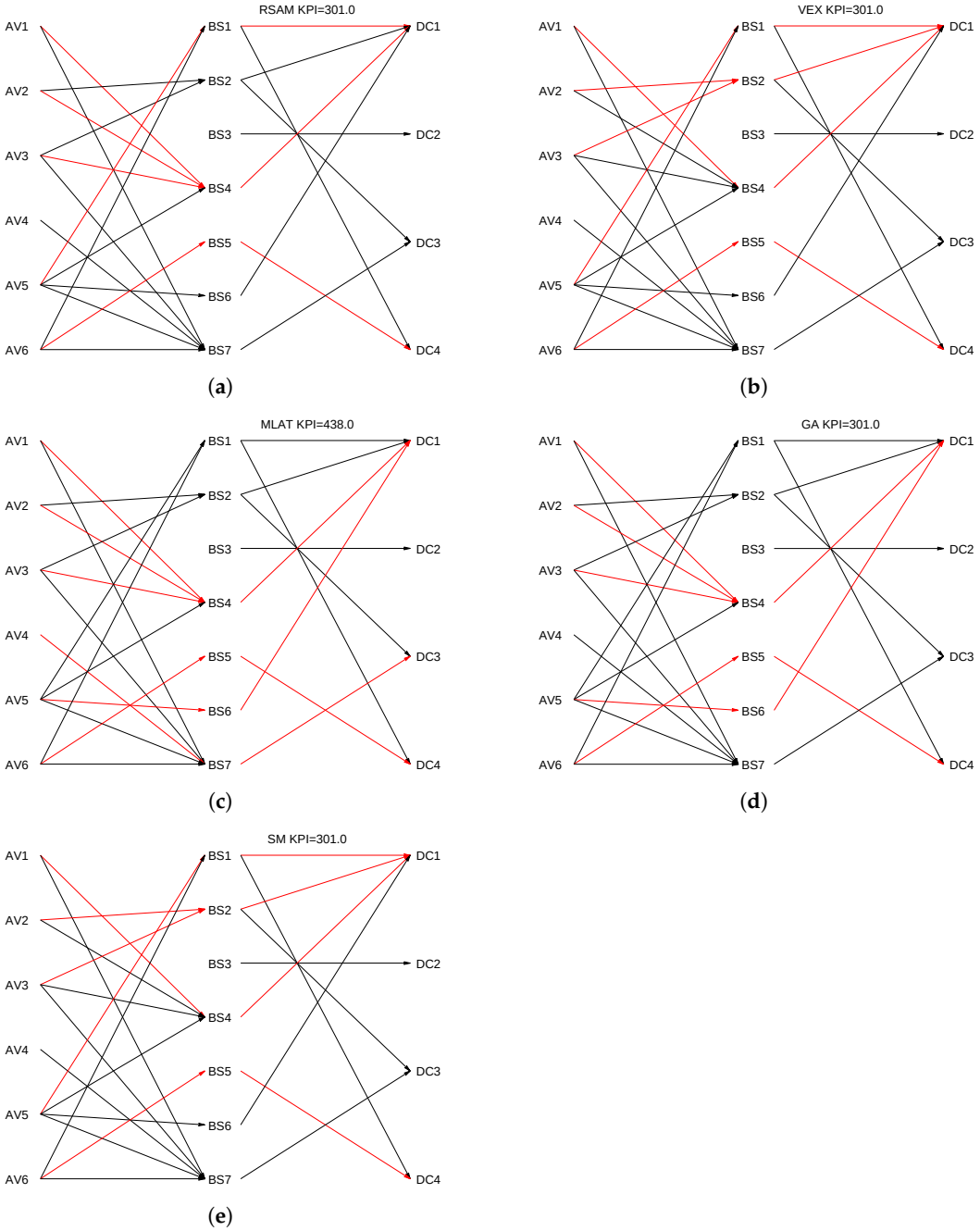


Figure 7. Paths in the graph corresponding to the found configurations for the *6/7/4_fixed* scenario: (**a**) RSAM solution. (**b**) VEX solution. (**c**) MLAT solution. (**d**) GA solution. (**e**) SM solution.

The configuration summaries printed in Figure 8a–e describe the plots of Figure 7a–e: the first line reports the algorithm and the found optimal KPI value, while the remaining lines detail each AV by reporting the connection BS and offloading DC, together with their respective loads, closed by the obtained latency value for the single AV. The results highlight that MLAT loads BS7 at 600%, thus resulting in a total latency of 438.0 ms, which is 46% higher than the minimum obtainable value. Note that this underperformance is due to MLAT minimizing each AV route to the DC independently from that of other AVs.

```
RSAM KPI=301.0
AV1 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV2 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV3 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV4 : internal = 111.0
AV5 : BS1  (0.02) --> DC1  (1.00) = 38.0
AV6 : BS5  (0.01) --> DC4  (0.25) = 38.0
```

(**a**)

```
VEX KPI=301.0
AV1 : BS4  (0.01) --> DC1  (1.00) = 38.0
AV2 : BS2  (0.03) --> DC1  (1.00) = 38.0
AV3 : BS2  (0.03) --> DC1  (1.00) = 38.0
AV4 : internal = 111.0
AV5 : BS1  (0.02) --> DC1  (1.00) = 38.0
AV6 : BS5  (0.01) --> DC4  (0.25) = 38.0
```

(**b**)

```
MLAT KPI=438.0
AV1 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV2 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV3 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV4 : BS7  (6.00) --> DC3  (0.12) = 248.0
AV5 : BS6  (0.04) --> DC1  (1.00) = 38.0
AV6 : BS5  (0.01) --> DC4  (0.25) = 38.0
```

(**c**)

```
GA KPI=301.0
AV1 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV2 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV3 : BS4  (0.04) --> DC1  (1.00) = 38.0
AV4 : internal = 111.0
AV5 : BS6  (0.04) --> DC1  (1.00) = 38.0
AV6 : BS5  (0.01) --> DC4  (0.25) = 38.0
```

(**d**)

```
SM KPI=301.0
AV1 : BS4  (0.01) --> DC1  (1.00) = 38.0
AV2 : BS2  (0.03) --> DC1  (1.00) = 38.0
AV3 : BS2  (0.03) --> DC1  (1.00) = 38.0
AV4 : internal = 111.0
AV5 : BS1  (0.02) --> DC1  (1.00) = 38.0
AV6 : BS5  (0.01) --> DC4  (0.25) = 38.0
```

(**e**)

Figure 8. Configurations found for the *6/7/4_fixed* scenario: (**a**) RSAM solution. (**b**) VEX solution. (**c**) MLAT solution. (**d**) GA solution. (**e**) SM solution.

5.2. 10/25/13_Fixed Scenario

Scenario *10/25/13_fixed* is more complex, as it has $N_{\mathrm{AV}} = 10$, $N_{\mathrm{BS}} = 25$, and $N_{\mathrm{DC}} = 13$ (Figure 9). Note that three BSs are deliberately not connected to any AV, as it may happen in a real case; nonetheless, they matter in the optimal configuration search. The service in each DC has typical latency $l_m = 33$ ms and typical load $\gamma_m = 1/8$ CPU. The internal service latency is still $\ell_i = 111$ ms, while the bit rates produced by the AVs vary from 20 Mbit/s to 31 Mbit/s. In practice, each BS offloads to a single DC, and each DC serves two BSs with low-latency wired links, emulating the case of IEEE 802.11ad access points with edge computers. There is a single station (BS25) that is connected with high latency to a single node (DC13), emulating the case of 5G radio access and a remote cloud server. For this scenario, the number of possible configurations is $N_{\mathrm{all}} = 1.4 \times 10^{25}$, of which $N_{\mathrm{val}} = 1.6 \times 10^{8}$ are valid.

KPIs for the found solutions are reported in Table 4. This time, VEX cannot be used as the real optimum due to its complexity. However, RSAM, MLAT, GA, and SM all agree and find an optimal solution, achieving an average latency of 38 ms, a radio load of 1%, a DC load of 10%, and no radio or CPU overloads. The four obtained solutions are different, but all of them reach the minimum latency. The complexity and run time results show that GA is the slowest and most complex, while MLAT is the quickest. RSAM and SM are quick too, with times of 649 ms and 11 ms, respectively.

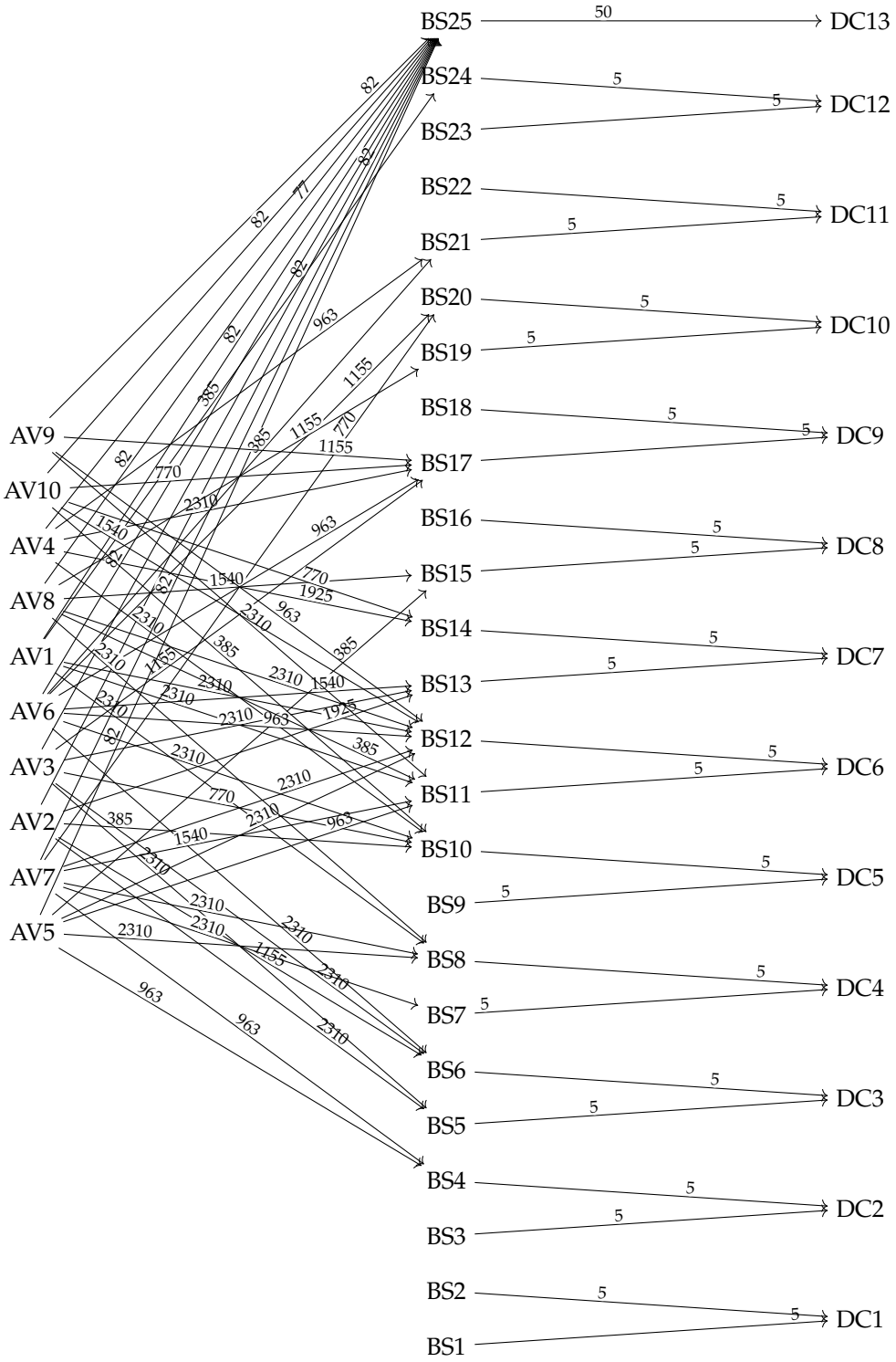
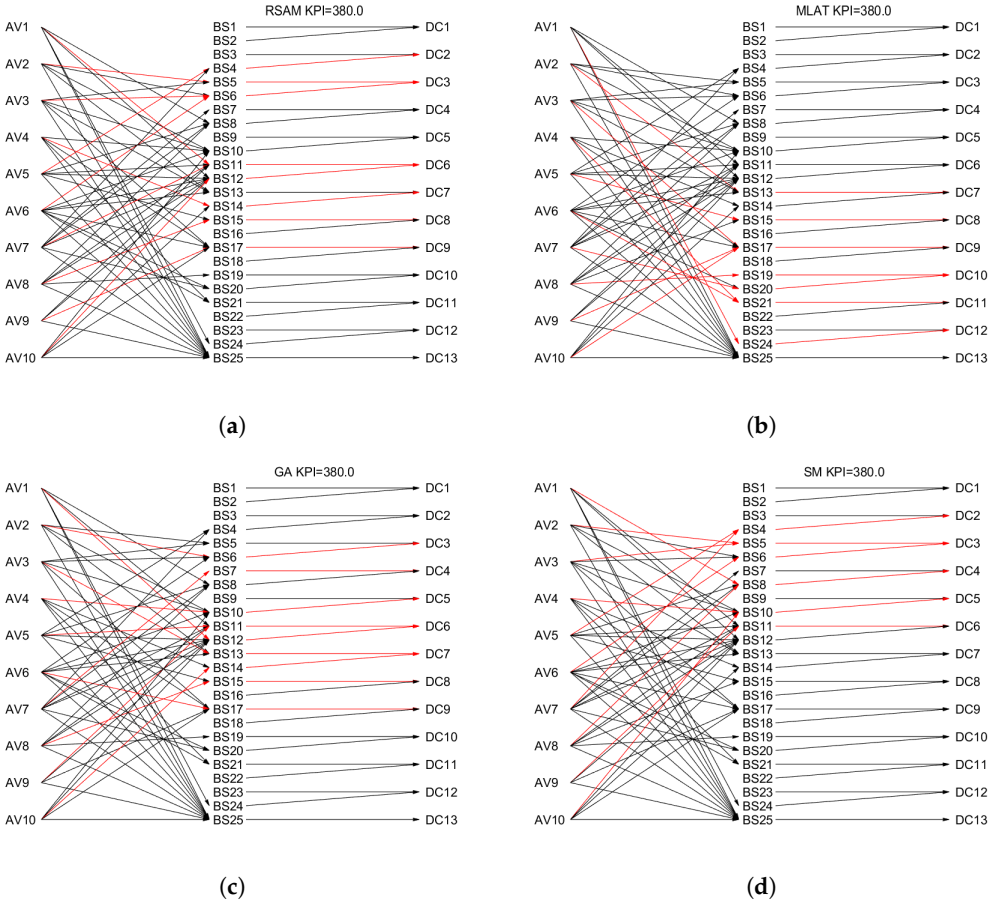


Figure 9. Connection graph of the *10/25/13_fixed* scenario.

Table 4. KPIs for the solutions to the *10/25/13_fixed* scenario.

	RSAM	**MLAT**	**GA**	**SM**
$\sum_i L_i$ (ms)	380.0	380.0	380.0	380.0
$\frac{1}{N_{\mathrm{AV}}} \sum_i L_i$ (ms)	38.0	38.0	38.0	38.0
$\frac{1}{N_{\mathrm{BS}}} \sum_k U_k$	0.01	0.01	0.01	0.01
V_{BS}	0	0	0	0
$\frac{1}{N_{\mathrm{DC}}} \sum_m \Gamma_m$	0.10	0.10	0.10	0.10
V_{DC}	0	0	0	0
N_{call}	50,000	1	953,440	161
Run time (s)	0.649	0.005	21.469	0.011

The algorithms produce the graphs shown in Figure 10a–d, from which we see that all algorithms tend to not overload the DCS by spreading the AVs through many computation nodes.



(**a**) (**b**)

(**c**) (**d**)

Figure 10. Paths in the graph corresponding to the found configurations for the *10/25/13_fixed* scenario: (**a**) RSAM solution. (**b**) MLAT solution. (**c**) GA solution. (**d**) SM solution.

The configuration summaries corresponding to Figure 10a–d are printed in Figure 11a–d.

```
RSAM KPI=380.0
AV1 : BS11 (0.03) --> DC6  (0.38) = 38.0
AV2 : BS5  (0.01) --> DC3  (0.38) = 38.0
AV3 : BS6  (0.02) --> DC3  (0.38) = 38.0
AV4 : BS14 (0.01) --> DC7  (0.12) = 38.0
AV5 : BS4  (0.03) --> DC2  (0.12) = 38.0
AV6 : BS6  (0.02) --> DC3  (0.38) = 38.0
AV7 : BS11 (0.03) --> DC6  (0.38) = 38.0
AV8 : BS15 (0.02) --> DC8  (0.12) = 38.0
AV9 : BS17 (0.02) --> DC9  (0.12) = 38.0
AV10: BS12 (0.01) --> DC6  (0.38) = 38.0
```

(**a**)

```
MLAT KPI=380.0
AV1 : BS24 (0.05) --> DC12 (0.12) = 38.0
AV2 : BS13 (0.01) --> DC7  (0.12) = 38.0
AV3 : BS17 (0.07) --> DC9  (0.38) = 38.0
AV4 : BS21 (0.08) --> DC11 (0.25) = 38.0
AV5 : BS15 (0.08) --> DC8  (0.12) = 38.0
AV6 : BS21 (0.08) --> DC11 (0.25) = 38.0
AV7 : BS20 (0.04) --> DC10 (0.25) = 38.0
AV8 : BS19 (0.02) --> DC10 (0.25) = 38.0
AV9 : BS17 (0.07) --> DC9  (0.38) = 38.0
AV10: BS17 (0.07) --> DC9  (0.38) = 38.0
```

(**b**)

```
GA KPI=380.0
AV1 : BS12 (0.01) --> DC6  (0.38) = 38.0
AV2 : BS6  (0.02) --> DC3  (0.12) = 38.0
AV3 : BS13 (0.01) --> DC7  (0.25) = 38.0
AV4 : BS10 (0.01) --> DC5  (0.12) = 38.0
AV5 : BS11 (0.04) --> DC6  (0.38) = 38.0
AV6 : BS17 (0.02) --> DC9  (0.12) = 38.0
AV7 : BS7  (0.01) --> DC4  (0.12) = 38.0
AV8 : BS15 (0.02) --> DC8  (0.12) = 38.0
AV9 : BS11 (0.04) --> DC6  (0.38) = 38.0
AV10: BS14 (0.03) --> DC7  (0.25) = 38.0
```

(**c**)

```
SM KPI=380.0
AV1 : BS8  (0.02) --> DC4  (0.25) = 38.0
AV2 : BS5  (0.02) --> DC3  (0.38) = 38.0
AV3 : BS5  (0.02) --> DC3  (0.38) = 38.0
AV4 : BS10 (0.07) --> DC5  (0.25) = 38.0
AV5 : BS4  (0.06) --> DC2  (0.25) = 38.0
AV6 : BS6  (0.01) --> DC3  (0.38) = 38.0
AV7 : BS4  (0.06) --> DC2  (0.25) = 38.0
AV8 : BS8  (0.02) --> DC4  (0.25) = 38.0
AV9 : BS11 (0.01) --> DC6  (0.12) = 38.0
AV10: BS10 (0.07) --> DC5  (0.25) = 38.0
```

(**d**)

Figure 11. Configurations found for the *10/25/13_fixed* scenario: (**a**) RSAM solution. (**b**) MLAT solution. (**c**) GA solution. (**d**) SM solution.

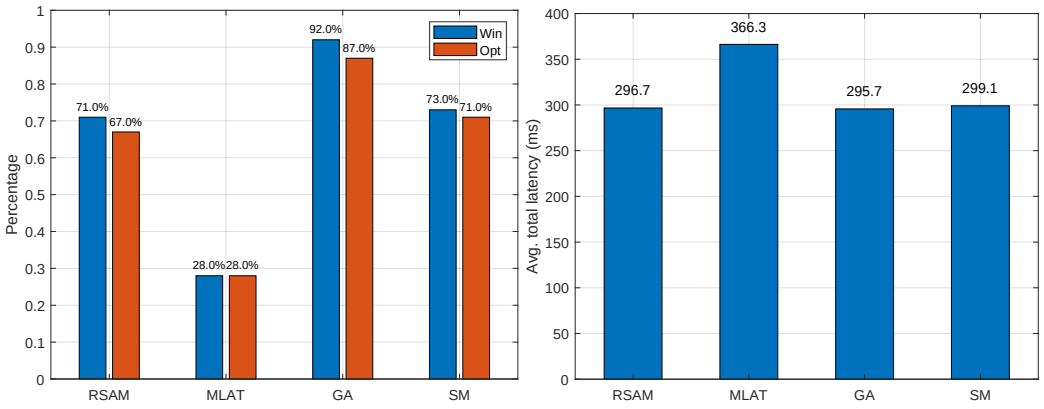
5.3. Variable Scenarios

Scenario *6/7/4_variable* has the same complexity of *6/7/4_fixed*, but UL access rates are generated randomly with uniform distribution between 0 Mbit/s and 20 Mbit/s and wired link delays randomly with uniform distribution between 5 ms and 150 ms. The connection matrices are created in this way: UL access rates lower than 9 Mbit/s are considered to not provide connection, and link delays higher than 60 ms correspond to an unavailable link. As for the other parameters, $\rho_i = 5$ Mbit/s, $\gamma_m = 1/4$ CPU, $l_m = 33$ ms, and $\ell_i = 100$ ms. Differently from the fixed case, every simulation consists of 100 randomly generated configurations, for which the performance of all algorithms is averaged for comparison. For this scenario, on average, there are $N_{\text{val}} = 2.1 \times 10^5$ valid configurations, so we consider VEX as the optimum, reference solution.

Figure 12a shows the percentage of times an algorithm achieves the best KPI among the four algorithms (ties are counted as wins), and its KPI equals that of the VEX solution. GA wins in 92% of cases, whereas SM does so in 73 % of cases and RSAM in 71% of cases. MLAT is the last, winning only in 28% of cases. However, while GA, SM, and RSAM do not guarantee to obtain the optimum value in all victory situations, MLAT does. Although there is a 19% difference between GA and SM, the loss in terms of average total latency is minimal (Figure 12b): GA achieves a value of 295.7 ms, SM obtains 299.1 ms, and RSAM is slightly better, at 296.7 ms, while MLAT underperforms at 366.3 ms. VEX provides the optimal value of 295.0 ms.

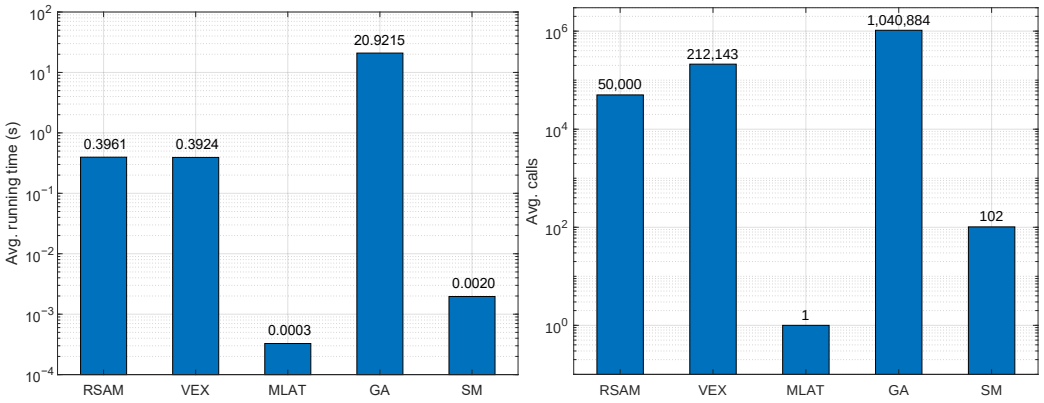
The moderate KPI loss of SM with respect to GA, however, is compensated by a significant running time reduction, shown in Figure 13a. While SM runs in 2 ms, GA takes 20.9 s, and both RSAM and VEX employ 0.4 s. Figure 13b shows the number of calls, which is fixed for RSAM and MLAT, about 100 for SM, and more than 2×10^5 for VEX; GA instead peaks at about 10^6 calls.

Another scenario, *10/25/5_variable*, has nearly the same dimensions of *10/25/13_fixed*, except for the DCs, which are decreased to $N_{\text{DC}} = 5$ to simulate the effects of scarce computation resources; UL rates, link delays, connection matrixes, and AV bit rates are generated randomly as in scenario *6/7/4_variable*. Simulation results are averaged over 100 randomly generated configurations.



(**a**) (**b**)

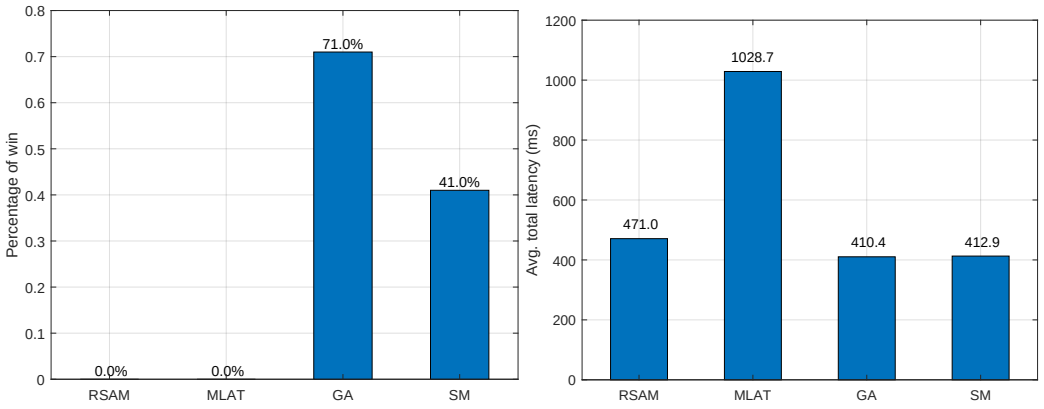
Figure 12. Scenario *6/7/4_variable*. (**a**) Percentage of winning and achieving optimum. (**b**) Average KPI.



(**a**) (**b**)

Figure 13. Scenario *6/7/4_variable*: (**a**) Average running time. (**b**) Average number of calls.

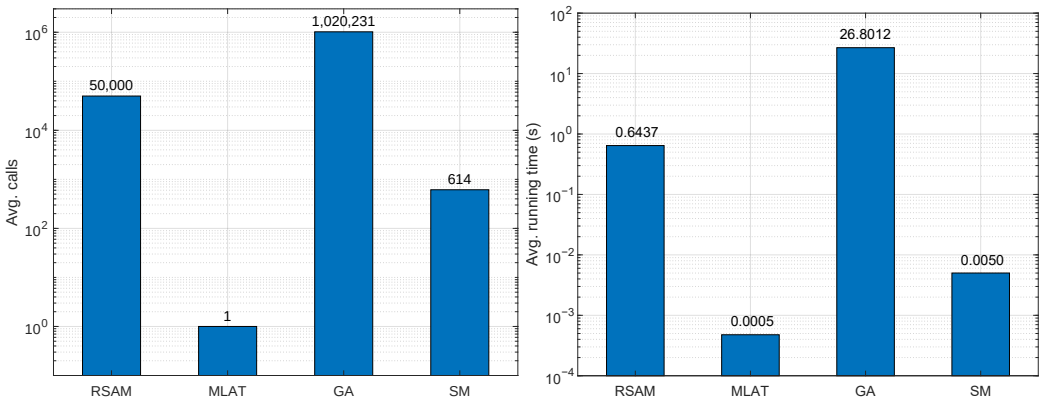
From Figure 14a, the percentage of winning is 71% and 41% for GA and SM, respectively, while RSAM and MLAT never win. Additionally, the difference in average total latency between GA and SM is small (410.4 ms for GA and 412.9 ms for SM), and RSAM average latency is sensibly the worst, while MLAT average latency is more than doubled with respect to GA (Figure 14b).



(**a**) (**b**)

Figure 14. Scenario *10/25/5_variable*: (**a**) Probability of win. (**b**) Average of the KPI.

As concerns the number of calls (Figure 15a) and the running times (Figure 15b), GA is still the slowest and most complex, with a 26.8 s run time, while RSAM takes 0.6 s, SM 5 ms, and MLAT less than 1 ms.



(**a**) (**b**)

Figure 15. Scenario *10/25/5_variable*: (**a**) Average number of calls. (**b**) Running time.

Finally, we introduce another variable scenario to compare the convergence performance of the best algorithm, GA, with that of the proposed SM, which is faster and only slightly less optimal. For this comparison, we consider the scenario *40/60/20_variable* with dimensions $N_{\mathrm{AV}} = 40$, $N_{\mathrm{BS}} = 60$, and $N_{\mathrm{DC}} = 20$. The fixed parameter values are $l_m = 33\,\mathrm{ms}$, $\gamma_m = 1/2$ CPU, $\ell_i = 111\,\mathrm{ms}$, while the bit rates produced by the AVs are random samples from a Gaussian distribution with mean 25 Mbit/s, standard deviation 7 Mbit/s, and bounded in the $[15, 35]$ Mbit/s interval (samples outside the interval are assigned the value of the nearest limit). The UL access rates are uniformly distributed in $[0, 120]$ Mbit/s, and the wired delays are uniformly distributed in $[5, 150]$ ms. BS connections are considered absent if the UL bit rate is lower than 25 Mbit/s, and DC connections are not kept if the delay is higher than 60 ms. Even in this case, 100 random configurations are generated, and the results are averaged. With these parameters, we obtain, on average, $\tilde{N}_{\mathrm{BS}} = 53.9$ and $\tilde{N}_{\mathrm{DC}} = 12.8$. Figure 16 shows the variation in the average total latency as the number of calls increases up to $N_{\mathrm{call}} = 10^6$ for the GA and SM algorithms. Initially, SM underperforms with respect to GA, up until 10^4 calls; after this point, SM improves the KPI and achieves its minimum before 2×10^4 calls. GA, on the other hand, achieves a slightly lower KPI after this point, but it needs almost two orders of magnitude more calls.

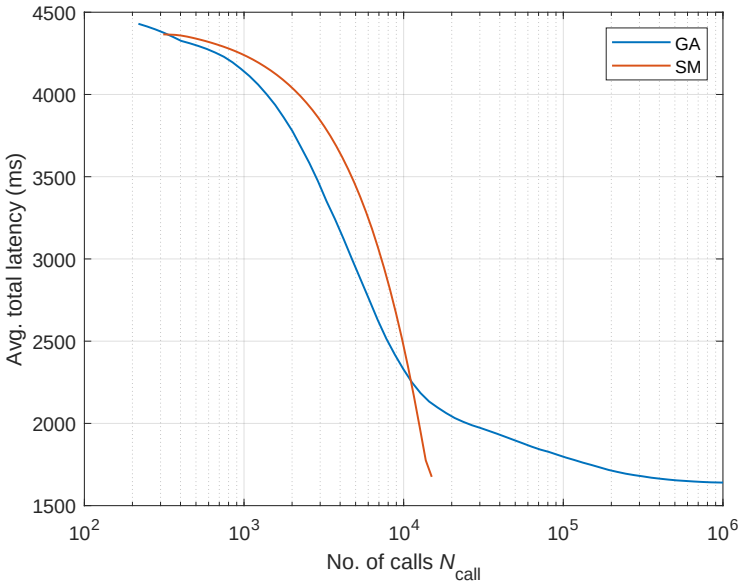


Figure 16. Scenario *40/60/20_variable*: achieved KPI as the number of calls varies.

5.4. Scalability

The capability to withstand the increase in problem dimensions is directly related to the algorithms computational cost, which is summarized in Table 5.

Table 5. Computational cost of the algorithms, ordered with decreasing complexity.

Algorithm	Complexity	Time for *40/60/20_Variable* (s)
VEX	$\mathcal{O}\left(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}} \cdot (\tilde{N}_{\mathrm{BS}} \tilde{N}_{\mathrm{DC}})^{N_{\mathrm{AV}}}\right)$	2×10^{100} (estimated)
GA	$\mathcal{O}\left(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}} \cdot N_{\mathrm{call}}\right)$	109.4
RSAM	$\mathcal{O}\left(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}} \cdot N_{\mathrm{call}}\right)$	4.5
SM	$\mathcal{O}\left(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}} \cdot \vartheta N_{\mathrm{AV}} \tilde{N}_{\mathrm{BS}} \tilde{N}_{\mathrm{DC}}\right)$	0.0356
MLAT	$\mathcal{O}\left(N_{\mathrm{AV}} N_{\mathrm{BS}} N_{\mathrm{DC}}\right)$	0.0028

Due to its exponential complexity, VEX is the least scalable approach, as it may become intractable even for moderately sized systems.

GA and RSAM have a complexity that is linear in the system size, but their cost greatly depends on the number of calls N_{call}. In our simulations, we considered $N_{\mathrm{call}} = 10^6$ for GA and $N_{\mathrm{call}} = 50,000$ for RSAM, since GA is used as a reference to find the optimal value, while RSAM could be adopted in a practical case. Vertical scalability can be guaranteed by keeping N_{call} limited and fixed as the problem dimensions increase so as to quickly find a solution; at the same time, this also causes a loss of the solution quality, since a progressively lower percentage of configurations are tested.

Instead, a nearly quadratic complexity results for SM, while practically less than linear complexity is assigned to MLAT, which is the least-costing one. Thus, both SM and MLAT are the most scalable.

Table 5 lists the obtained or estimated running times for the scenario *40/60/20_variable*; VEX is estimated to employ 2×10^{100} s, while all the other algorithms have reasonable running times. Specifically, GA takes 109.4 s, RSAM needs 4.5 s, and SM only 0.0356 s. MLAT is the fastest, with 0.0028 s.

In summary, the proposed algorithm SM has running times of a few ms for all the considered problem dimensions, achieves solutions with only slight loss from the optimal ones, and the KPIs are better than that of RSAM and MLAT.

6. Conclusions

This work has presented some novel algorithms to deal with the unique assignment of radio and CPU resources to a fleet of mobile robots needing task offloading. The assignment is performed to optimize an indicator of the operational efficiency of the robot fleet, such as the average response delay from the offloading servers. Mobile robots can exploit the assigned resources and perform their task in the quickest way possible. We have compared brute force, evolutionary, and reduced complexity search algorithms to find the optimal assignment. After defining a number of scenarios, we have simulated and compared the algorithms' performance in terms of complexity and optimality. Our results show that the proposed simplified algorithm (SM) outperforms the other methods in terms of reduced complexity, with only a slight average latency penalty with respect to the best algorithm (GA).

Author Contributions: Conceptualization, G.B. and L.R.; methodology, G.B. and L.R.; software, G.B.; validation, G.B. and L.R.; formal analysis, G.B. and L.R.; investigation, G.B. and L.R.; writing—original draft preparation, G.B. and L.R.; writing—review and editing, G.B. and L.R.; visualization, G.B. and L.R.; supervision, G.B. and L.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the European Union—Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP E83C22004640001, partnership on "Telecommunications of the Future" (PE00000001-program "RESTART").

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial intelligence
AV	Autonomous vehicle
BS	Base station
CPU	Central processing unit
CRAN	Cloud radio access network
DC	Data center
DCA	Difference of convex algorithm
GA	Genetic algorithm
HW	Hardware
KPI	Key performance indicator
MIMO	Multiple-input multiple-output
MBNLP	Mixed-binary nonlinear programming
MINLP	Mixed-integer nonlinear programming
MLAT	Minimum latency
NP	Nondeterministic polynomial
OS	Operating system
RSAM	Random sampling
SM	Sequential minimization
SVC	Service
UL	Uplink
VEX	Valid exhaustive

References

1. Ross, P.E. Robot, you can drive my car. *IEEE Spectrum* **2014**, *51*, 60–90. [CrossRef]
2. Nikitas, A.; Michalakopoulou, K.; Njoya, E.T.; Karampatzakis, D. Artificial Intelligence, Transport and the Smart City: Definitions and Dimensions of a New Mobility Era. *Sustainability* **2020**, *12*, 2789. [CrossRef]
3. Kuru, K.; Khan, W. A Framework for the Synergistic Integration of Fully Autonomous Ground Vehicles with Smart City. *IEEE Access* **2021**, *9*, 923–948. [CrossRef]
4. Xidias, E.; Zacharia, P.; Nearchou, A. Intelligent fleet management of autonomous vehicles for city logistics. *Appl. Intell.* **2022**, *52*, 18030–18048. [CrossRef]
5. Sarker, V.K.; Peña Queralta, J.; Gia, T.N.; Tenhunen, H.; Westerlund, T. Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–6. [CrossRef]
6. Qingqing, L.; Yuhong, F.; Queralta, J.P.; Gia, T.N.; Tenhunen, H.; Zou, Z.; Westerlund, T. Edge Computing for Mobile Robots: Multi-Robot Feature-Based Lidar Odometry with FPGAs. In Proceedings of the 2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU), Kathmandu, Nepal, 4–6 November 2019; pp. 1–2. [CrossRef]
7. Thong Tran, T.; Zhang, Y.C.; Liao, W.T.; Lin, Y.J.; Li, M.C.; Huang, H.S. An autonomous Mobile Robot System based on Serverless Computing and Edge Computing. In Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Republic of Korea, 22–25 September 2020; pp. 334–337. [CrossRef]
8. Ahmed Abdulsaheb, J.; Jasim Kadhim, D. Real-Time SLAM Mobile Robot and Navigation Based on Cloud-Based Implementation. *J. Robot.* **2023**, *2023*, 9967236. [CrossRef]
9. Vinod, D.; Singh, D.; Saikrishna, P.S. Data-Driven MPC for a Fog-Cloud Platform With AI-Inferencing in Mobile-Robotics. *IEEE Access* **2023**, *11*, 99589–99606. [CrossRef]

10. Baruffa, G.; Detti, A.; Rugini, L.; Crocetti, F.; Banelli, P.; Costante, G.; Valigi, P. AI-Driven Ground Robots: Mobile Edge Computing and mmWave Communications at Work. *IEEE Open J. Commun. Soc.* **2024**, *5*, 3104–3119. [CrossRef]
11. Lin, K.C.J.; Wang, H.C.; Lai, Y.C.; Lin, Y.D. Communication and Computation Offloading for Multi-RAT Mobile Edge Computing. *IEEE Wirel. Commun.* **2019**, *26*, 180–186. [CrossRef]
12. Chen, J.; Zhang, X.; Xin, B.; Fang, H. Coordination between unmanned aerial and ground vehicles: A taxonomy and optimization perspective. *IEEE Trans. Cybern.* **2015**, *46*, 959–972. [CrossRef] [PubMed]
13. Maheshwari, S.; Raychaudhuri, D.; Seskar, I.; Bronzino, F. Scalability and Performance Evaluation of Edge Cloud Systems for Latency Constrained Applications. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 286–299. [CrossRef]
14. Zhou, Y.; Yu, F.R.; Chen, J.; He, B. Joint Resource Allocation for Ultra-Reliable and Low-Latency Radio Access Networks with Edge Computing. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 444–460. [CrossRef]
15. Baruffa, G.; Rugini, L.; Frescura, F.; Banelli, P. Radio and Computation Resource Management in Unmanned Vehicles with Edge Computing. In Proceedings of the International Conference on Computing, Networking, and Communication (ICNC 2025), Honolulu, HI, USA, 17–20 February 2025.
16. Sardellitti, S.; Barbarossa, S.; Scutari, G. Distributed mobile cloud computing: Joint optimization of radio and computational resources. In Proceedings of the 2014 IEEE Globecom Workshops (GC Wkshps), Austin, TX, USA, 8–12 December 2014; pp. 1505–1510. [CrossRef]
17. Sardellitti, S.; Scutari, G.; Barbarossa, S. Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Trans. Signal Inf. Process. Over Netw.* **2015**, *1*, 89–103. [CrossRef]
18. Eriksson, E.; Dan, G.; Fodor, V. Radio and Computational Resource Management for Fog Computing Enabled Wireless Camera Networks. In Proceedings of the 2016 IEEE Globecom Workshops (GC Wkshps), Washington, DC, USA, 4–8 December 2016; pp. 1–6. [CrossRef]
19. Zhao, P.; Tian, H.; Qin, C.; Nie, G. Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing. *IEEE Access* **2017**, *5*, 11255–11268. [CrossRef]
20. Gu, Y.; Chang, Z.; Pan, M.; Song, L.; Han, Z. Joint Radio and Computational Resource Allocation in IoT Fog Computing. *IEEE Trans. Veh. Technol.* **2018**, *67*, 7475–7484. [CrossRef]
21. Liu, Q.; Han, T.; Ansari, N. Joint Radio and Computation Resource Management for Low Latency Mobile Edge Computing. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–7. [CrossRef]
22. Wang, P.; Yao, C.; Zheng, Z.; Sun, G.; Song, L. Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems. *IEEE Internet Things J.* **2019**, *6*, 2872–2884. [CrossRef]
23. Chen, H.; Zhao, D.; Chen, Q.; Chai, R. Joint Computation Offloading and Radio Resource Allocations in Small-Cell Wireless Cellular Networks. *IEEE Trans. Green Commun. Netw.* **2020**, *4*, 745–758. [CrossRef]
24. Zhao, L.; Wu, D.; Zhou, L.; Qian, Y. Radio Resource Allocation for Integrated Sensing, Communication, and Computation Networks. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 8675–8687. [CrossRef]
25. Bouhoula, S.; Avgeris, M.; Leivadeas, A.; Lambadaris, I. DRL-based Trajectory Planning and Sensor Task Scheduling for Edge Robotics. In Proceedings of the 2024 IEEE 10th World Forum on Internet of Things (WF-IoT), Ottawa, ON, Canada, 10–13 November 2024; pp. 352–357. [CrossRef]
26. Ren, Y.; Friderikos, V. Pareto Optimal Task Offloading and Mobile Robots Paths in Edge Cloud Assisted mmWave Networks. In Proceedings of the 2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Valencia, Spain, 2–5 September 2024; pp. 1–6. [CrossRef]
27. Shu, J.; Ota, K.; Dong, M. RIS-Enabled Integrated Sensing, Computing, and Communication for Internet of Robotic Things. *IEEE Internet Things J.* **2024**, *11*, 32503–32513. [CrossRef]
28. Huynh, D.V.; Khosravirad, S.R.; Cotton, S.L.; Vu, T.X.; Dobre, O.A.; Shin, H.; Duong, T.Q. Joint Sensing, Communications, and Computing Design for 6G URLLC Service-Oriented MEC Networks. *IEEE Internet Things J.* **2024**, *11*, 32429–32439. [CrossRef]
29. Peng, M.; Zhang, K.; Jiang, J.; Wang, J.; Wang, W. Energy-Efficient Resource Assignment and Power Allocation in Heterogeneous Cloud Radio Access Networks. *IEEE Trans. Veh. Technol.* **2015**, *64*, 5275–5287. [CrossRef]
30. Luong, P.; Gagnon, F.; Despins, C.; Tran, L.N. Joint Virtual Computing and Radio Resource Allocation in Limited Fronthaul Green C-RANs. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 2602–2617. [CrossRef]
31. Ferdouse, L.; Anpalagan, A.; Erkucuk, S. Joint Communication and Computing Resource Allocation in 5G Cloud Radio Access Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 9122–9135. [CrossRef]
32. Mharsi, N.; Hadji, M. Edge computing optimization for efficient RRH-BBU assignment in cloud radio access networks. *Comput. Netw.* **2019**, *164*, 106901. [CrossRef]

33. Shirzad, F.; Ghaderi, M. Joint Computing and Radio Resource Allocation in Cloud Radio Access Networks. In Proceedings of the 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS), Virtual, 4–7 October 2021; pp. 518–526. [CrossRef]
34. Wong, C.S.; Tan, I.; Kumari, R.D.; Wey, F. Towards achieving fairness in the Linux scheduler. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 34–43. [CrossRef]
35. Andradóttir, S. An overview of simulation optimization via random search. In *Handbooks in Operations Research and Management Science*; Elsevier: Amsterdam, The Netherlands, 2006; Volume 13, pp. 617–631.
36. Yang, X. *Engineering Optimization: An Introduction with Metaheuristic Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
37. Sastry, K.; Goldberg, D.; Kendall, G. Genetic Algorithms. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*; Springer: Boston, MA, USA, 2005; pp. 97–125. [CrossRef]
38. Nitsche, T.; Cordeiro, C.; Flores, A.B.; Knightly, E.W.; Perahia, E.; Widmer, J.C. IEEE 802.11ad: Directional 60 GHz communication for multi-Gigabit-per-second Wi-Fi. *IEEE Commun. Mag.* **2014**, *52*, 132–141. [CrossRef]
39. Lin, X.; Li, J.; Baldemair, R.; Cheng, J.F.T.; Parkvall, S.; Larsson, D.C.; Koorapaty, H.; Frenne, M.; Falahati, S.; Grovlen, A.; et al. 5G New Radio: Unveiling the Essentials of the Next Generation Wireless Access Technology. *IEEE Commun. Stand. Mag.* **2019**, *3*, 30–37. [CrossRef]
40. Qiu, T.; Chi, J.; Zhou, X.; Ning, Z.; Atiquzzaman, M.; Wu, D.O. Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2462–2488. [CrossRef]
41. Michalopoulou, M.; Kolios, P.; Panayiotou, T.; Ellinas, G. Meal delivery services: Current practices, challenges, and future directions. *IEEE Potentials* **2024**, *43*, 20–27. [CrossRef]
42. Donati, L.; Fontanini, T.; Tagliaferri, F.; Prati, A. An Energy Saving Road Sweeper Using Deep Vision for Garbage Detection. *Appl. Sci.* **2020**, *10*, 8146. [CrossRef]