Reinforcement Learning for Security-Aware Computation Offloading in Satellite Networks

Sthapit, S, Lakshminarayana, S, He, L, Epiphaniou, G & Maple, C

DOI 10.1109/JIOT.2021.3135632
ISSN 2327-4662

Publisher: IEEE

Reinforcement Learning for Security Aware Computation Offloading in Satellite Networks

Saurav Sthapit, Subhash Lakshminarayana, Ligang He, Gregory Epiphaniou and Carsten Maple

Abstract—The rise of *NewSpace* provides a platform for small and medium businesses to commercially launch and operate satellites in space. In contrast to traditional satellites, *NewSpace* provides the opportunity for delivering computing platforms in space. However, computational resources within space are usually expensive and satellites may not be able to compute all computational tasks locally. Computation Offloading (CO), a popular practice in Edge/Fog computing, could prove effective in saving energy and time in this resource-limited space ecosystem. However, CO alters the threat and risk profile of the system. In this paper we analyse security issues in space systems and propose a security-aware algorithm for CO. Our method is based on the reinforcement learning technique, Deep Deterministic Policy Gradient (DDPG). We show, using Monte-Carlo simulations, that our algorithm is effective under a variety of environment and network conditions and provide novel insights into the challenge of optimised location of computation.

Index Terms—Computation Offloading, IOT, Cyber-Security, NewSpace, Reinforcement Learning, LEO satellites.

1 Introduction

The space industry is experiencing rapid growth currently, thanks to lowering technological and economic barriers to entry. Commercial Off-the-Shelf (COTS) hardware such as Nvidia Jetson [1] and Xilinx Virtex FPGA are readily available along with the plethora of software and support [1], [2], [3]. Similarly, Software Defined Networking (SDN) [4], [5] is continuing to revolutionise the way we connect, making it easier, more flexible and cheaper. Advances such as these have led to new commercial companies, including relatively Small and Medium Enterprises (SMEs), entering the space industry that has previously been restricted to large non-commercial organisations such as National Aeronautics and Space Administration (NASA). This new environment has been coined as the '*NewSpace*' [6]. The *NewSpace* ecosystem comprises of thousands of satellites of all sizes, and contrary to traditional satellites, the satellites such as Cubesats [7] can be as small as $10 \times 10 \times 10\ cm^3$. In this new paradigm, instead of acting solitary, the satellites may form a cluster or a constellation, communicate with each other, and jointly serve the users on the earth surface.

In terms of applications, satellites used to be limited to relaying information from one point to another. Such architecture is commonly referred as bent-pipe architecture [8]. However, modern satellites are intelligent; instead of being simple relays, they can sense, process and act intelligently [9]. For example, a satellite can autonomously collect space debris or dock itself without human intervention [10]. Satellites are also able to continuously monitor the environment using multiple sensors. In such cases, it is desirable to process the raw data in the orbit itself rather than transferring all of the data to Earth [11]. However, this extra computation will add to the existing sensing and processing of the sensor data and not all of the satellites may be able to handle them [12] due to limitations in energy and computational power.

Satellites, such as Cubesats will have to rely on other nearby satellites or space stations for processing their sensor data. Attempts have already been made to address these challenges. Recently, super computer satellites as small as a kitchen microwave are being launched in space [13]. The objective of such super computer satellites is to offer 'computing as a service' to other satellites in order to process the sensor data while in orbit. This process of offloading computation is already common in terrestrial computations for edge devices and is commonly referred to as Computation Offloading (CO). While CO in space is similar to CO on Earth in many respects, there exist some unique challenges. These include, (1) the server in space may not be as powerful as the server on Earth. This implies that there is a non-trivial queuing and computation delay at the server, which is not present in terrestrial applications. (2) The satellite network is very dynamic, especially in the Low Earth Orbit (LEO) orbit. Hence, the topology of the satellite network will be changing rapidly. (3) CO in space requires data to be transmitted to a different platform (satellite). This additional communication requirement will raise security risks, such as eavesdropping (from nearby satellites), data modification, and/or preventing the offloading satellite from accessing such service. Decisions regarding whether to offload and the level of security measures to be used in exchanging the data (between the server and the client satellite) are not trivial decisions. Careful consideration of the environment is required to assess if such offloading is beneficial in terms

- *S. Sthapit, G. Epiphaniou and C. Maple are with the Warwick Manufacturing Group, University of Warwick, UK*
- *S. Lakshminarayana is with the School of Engineering, University of Warwick, UK*
- *L. He is with the Department of Computer Science, University of Warwick, UK*

E-mail: { saurav.sthapit, subhash.lakshminarayana, ligang.he, gregory.epiphaniou, cm } @warwick.ac.uk

1. https://developer.nvidia.com/embedded/jetson-developer-kits

of time, energy, and the security risks incurred.

In this work, we explore CO in the context of satellites and *NewSpace* with the awareness of security threats in space. We formulate the security-aware CO problem as a multi-objective optimisation problem and jointly minimise the time, energy and security cost of the system using a Reinforcement Learning (RL) framework. Since our formulation involves decision variables that are continuous, we use the Deep Deterministic Policy Gradient (DDPG) method to solve the RL problem, as it can be directly applied to continuous action spaces, and avoids the need for discretisation [14]. Our results show that even in the presence of wireless communications security threats, it is possible to offload computation and increase the efficiency of the system. The main contributions of the paper are as follows:

- A new examination of the space landscape for communications security.
- The formulation of the security-aware CO problem within New space as a multi-objective problem.
- The development of a new DDPG-based solution to solve the problem and analysis of its efficacy in comparison to an state of the art Deep Q-Network (DQN) based solution.

We note that while there is extensive literature in wireless communications related to resource allocation/scheduling [15], [16], [17], security [18], [19], computational offloading [20], [21], [22], [23], [24], [25], etc., the focus of all these works is on terrestrial mobile networks. In contrast, our work considers inter-satellite communications while incorporating the aforementioned domain-specific features. To the best of our knowledge, this work is the first to consider security-aware offloading in a satellite environment, and this is one of our important contributions.

The paper is structured as follows. Section 2 describes the satellite architecture, inter-satellite communication and various security risks in space applications. In Section 3, we define the basics of CO including the local execution and remote execution. Section 4 presents an overview the system, formulates the problem and presents our solution. In Section 5, we present the experimental results. Finally, we conclude the paper in Section 6.

2 Satellite Architecture

In this section, we present a brief overview of the satellite architecture, the communication requirements, and present the current and future applications for satellite networks.

2.1 Constellations

Traditionally, satellites were designed to operate in a solitary environment and their data flow followed a bent-pipe architecture where an earth station transmits the data to the satellite in the uplink. The satellite amplifies the signal and transmits it to another Earth station in the downlink. An example is the usage of geo-stationary satellite for voice calls [26]. Due to their high altitude (36,000 km), the area covered by a geo-stationary satellite can be large. Hence, only a few of them are necessary to cover the entire earth. However, as their distance is large, the communication delay is large as well. Typical Round Trip Time (RTT) for a geo-stationary satellite can be more than 600 ms [27]. To minimise this delay, the satellites have to orbit the earth at a much lower altitude. However, this means only a fraction of the earth's region can be covered at any time and many satellites would be necessary for global coverage. For example, Iridium network system operating at 760 km (LEO) requires 66 satellites to cover the entire earth's surface [28]. Such a satellite formation can work solitary or in a constellation. According to [26], [29], there are three common types of formations of satellites namely:

- Trailing: In this formation, satellites share the same orbit, but are separated by a specific distance.
- Cluster/ Swarm: Satellites in this formation fly in close proximity to each other but in their own orbits.
- Constellation: A set of satellites organised in different orbital planes that cover the entire earth. Reference [30] presents a large scale constellation design framework for Internet of Space Things (IOST).

2.2 Communication

The wireless communication refers to the transmission and reception of the data between a satellite and other entities. A satellite may communicate with

1) other satellites and space station,
2) earth bound entities, and
3) planetary rovers.

The satellites may not only communicate to other satellites in the formations described above but also between the satellites in LEO, Medium Earth Orbit (mEO) and geo stationary (GEO) orbits [31], [32]. The reader may refer to [29], [31], [33], [34] for a comprehensive surveys on inter satellite communication system.

2.3 Computing in space

The small and nano satellite constellations can be used for various applications. For example, the satellite mesh can be used as a backhaul network providing high-speed, low latency communication links [9]. It could be particularly useful in remote areas where the terrestrial network does provide coverage. Reference [28] shows how the network load can be anticipated based on the geographical location and uses communication links between LEO satellites, as well as LEO and GEO satellites to improve the Quality of Service (QOS) for the end users in their communication needs.

The small and nano-satellites can provide a computational platform in space [35], [36]. It could process data generated by itself. For example, satellites and their networks can be used for sensing the earth. [9] referred to monitoring and reconnaissance capabilities of satellite networks as *"Eyes in the sky"*. Satellites equipped with sensors such as cameras can monitor the earth's surface 24×7. However, not all data that is acquired is useful due to various reasons. For example, if the interest is monitoring assets in an urban area, images of rural areas or oceans are of little interest. Also, if the images acquired are occluded by the clouds it could of little to no use [11]. Transmitting all the acquired images to the earth station can put a strain on the communication

links as well as on the earth station. The problem would only exacerbate in the future when more satellites are launched. If the satellites could pre-process the images and only retain and transmit the useful images, the resources could be more manageable. [11] used various deep learning algorithms to filter out the images that are occluded by the clouds.

In a different scenario, the authors in [37] describe using satellites to provide continuous Internet of Vehicles (IOV) services. In this case, the data generated by the vehicles on the surface of the earth can use edge-computing and communication services provided by the satellites to communicate with other vehicles in the IOV. The satellites not only act as a low latency communication medium but also a computing platform. In this work, we assume the satellites to be resource-limited computation platforms and the proposed algorithms would benefit such systems.

2.4 Security risks in Space

In space, attacks can be physical/kinetic or cyber [38], and the impact caused by such attacks depends on the sophistication of the attackers. If the attacker is an individual with limited capabilities like a hacktivist or an insider with limited capabilities, the impact may be limited. If the threat actor is a hostile nation-state or a privileged insider, the impact will be significantly higher [39]. Consider a defunct satellite that is out of service but still in orbit. If a hacker gains access to such a satellite, they may launch an attack on other satellites and services. The major reasons for concerns are the following:

- COTS hardware and software may have reported flaws and threats. Satellites may be in orbit for a long period of time (years) by which time new vulnerabilities could be discovered. It may be impossible or financially infeasible to apply the patch or update the software in the orbit,
- SMEs may overlook security in favour of cost-saving,
- hackers and activists also have access to the same technology (hardware and software) as it is readily available.

Reference Architecture (RA) is often used to understand and mitigate the security risks. They can be used in conjunction with attack trees for security-minded verification [10]. Figure 1 shows a functional RA of a satellite operating in an orbit. It shows the functional blocks within the satellite and interfaces for it to interact with the external world. It also highlights the attack surface of the satellite such as the Input/Output ports that may be targeted in an attack. In this work, however, we focus our study on the attacks that may be directed toward wireless communication. In general, cyber attacks affect one or more of the three aspects of security collectively known as the Confidentiality Integrity Availabilty (CIA) triad.

2.4.1 Confidentiality

Data confidentiality refers to the protection of transmitted data from passive attacks such as eavesdropping [40]. If confidential information is being shared without encryption or poor encryption, a passive attacker may listen to the communication or use data sniffing techniques to learn the victim's secrets [41]. Table 1 details various encryption algorithms ranked such that the strongest and slowest algorithm has the confidentiality score of one and other encryption algorithms are relative to it [42], [43], [44]. We assume the confidentiality score (S_{conf}) to be directly proportional to the process rate (i.e. stronger encryption algorithm has higher security overhead on the processor). In Section 5 we base our decision to select the appropriate encryption algorithm based on this table.

TABLE 1
Encryption algorithms in literature, their security level and process rate [45]

Encryption	Confidentiality (S_{conf})	Process Rate (Mb/s)
IDEA	1.0	11.76
DES	0.85	13.83
Blowfish	0.56	20.87
AES	0.53	22.03
RC4	0.32	37.17

TABLE 2
Different hashing algorithms, their security level and process rate [45]

Encryption	Integrity (S_{int})	Process Rate (Mb/s)
TIGER	1.0	75.76
RipeMD160	0.75	101.01
SHA-1	0.69	109.89
RipeMD128	0.63	119.05
MD5	0.44	172.41

2.4.2 Integrity

The integrity of the data is compromised when the attacker modifies the data from the sender to the receiver. Attacks such as the man-in-the-middle attack can modify the data, and the receiver satellite may have no knowledge about it. In an extreme scenario, if the receiving satellite has a propulsion system, the attacker may modify control messages to move away from their orbit, burn its fuel unnecessarily, or fatally crash with other nearby satellites [39]. Table 2 details a number of hashing algorithms to ensure the data has not been falsified. We used these values in Section 5 to select the appropriate hashing algorithm.

2.4.3 Availability

Attackers and hackers may try to disrupt the service provided (by the servers) by employing Denial Of Service (DOS) attacks or jamming the communication channel. Similar to attacks on the Earth's surface, attackers can affect the availability of wireless channels by transmitting at the same time as the legitimate satellite. Satellites sharing the same channel would be affected. Similarly, the attacker may target the server by making too many requests so that the legitimate node cannot be served. We simulate such attacks on availability in Section 5 by degrading the channel (–see Figure 2).

3 Computation Offloading

To understand and mitigate the attacks on communication systems of satellites and space systems, we study a process called Computation Offloading (CO). In Section 2.3, the application of satellites as a computing platform was

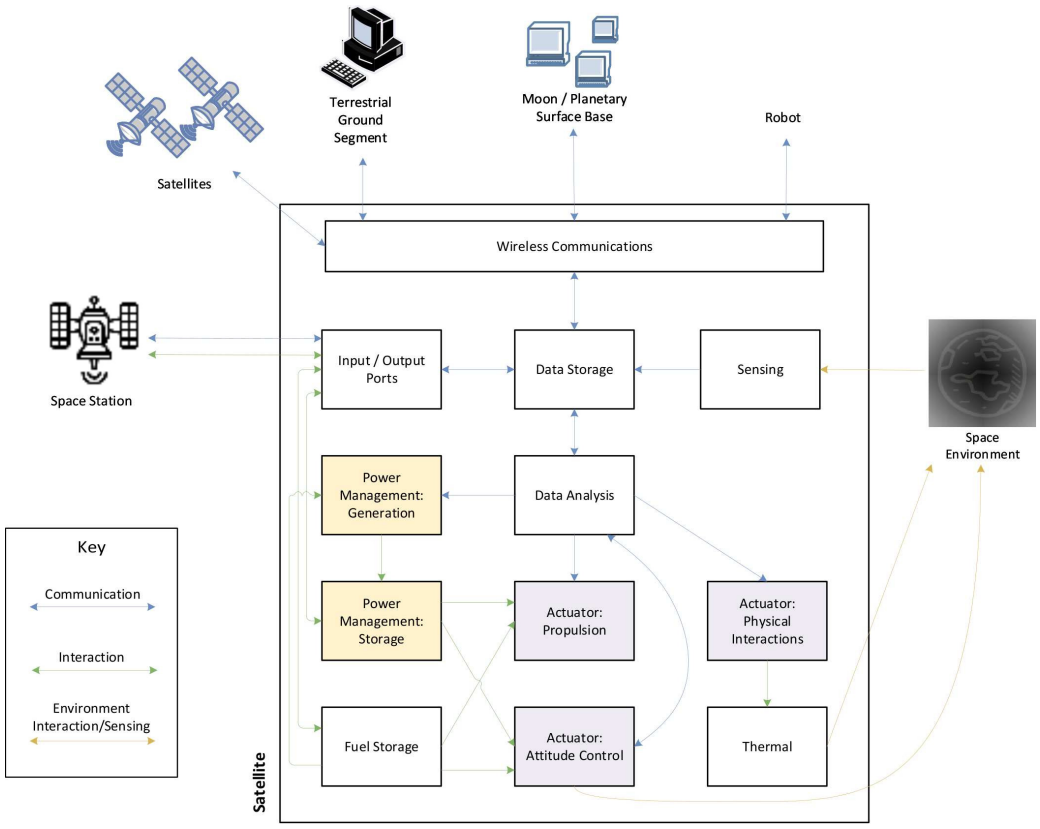

Fig. 1. Functional reference architecture of a satellite detailing its attack surface [39]

TABLE 3
List of important symbols.

D	Job data size.
E_{exec}	Total energy usage for executing the job.
E_{off}	Total energy usage for offloading the job.
I_m	Number of instructions a device can execute per second.
I_s	Number of instructions the server can execute per second.
J	Maximum number of jobs in an interval.
N_m	Number of servers in queuing model equals number of cores in the satellite.
N_s	Number of servers in queuing model equals number of cores in the space station.
P_m	Instantaneous power of satellite while executing the job.
R	Data Rate.
SD	Desired security level.
SP	Selected security level.
X	Number of instructions to execute a job per bit.
Λ	Job generation rate.
A	set of actions for the Markov Decision Process (MDP).
R	Reward function for the MDP.
S	set of states for the MDP.
T	Transition function for the MDP.
τ_d	Maximum allowed time for completion of a job.
τ_m	Time taken by satellite to execute the job.
r	Risk.
u	Central Processing Unit (CPU) utilisation.
μ	Service rate of the node.

described. However, in *NewSpace* systems, satellites may be constrained in terms of their computation and energy resource, CO will be very useful. CO is a process of delegating a computationally intensive task to an alternative device rather than on its own computing platform. This delegation may be done for achieving various goals such as improving latency, conserving energy, or both. Many CO algorithms have been proposed to offload algorithms from the edge devices to the cloud known as Mobile Cloud Computing (MCC) as well as to the edge servers known as Mobile Edge Computing (MEC) [20], [21], [22], [23], [24], [25]. In the context of *NewSpace*, a satellite may be considered as a resource-limited device that offloads some of its computation to the neighbouring satellites, space station or ground station to save resources. On the other hand, in the future, satellites may offer such computing services to User Equipment (UE) such as smartphones and vehicles, similar to services currently offered by cloud and edge servers. Especially with LEO satellites constellations such as OneWeb[2]. and Starlink[3] providing high-speed low-latency internet connectivity to worldwide coverage including remote areas not covered by cellular services.

Consider a resource-limited satellite such as Cubesats equipped with a sensor that is operating in the orbit. The sensor senses its surroundings and readings are sent to the processor periodically. The readings have to be processed by a computationally intensive algorithm which requires computing and energy resources. The satellite can, however, offload the computing to a nearby satellite or space station which has a significantly higher computational capability as well as energy resources. Also, for simplicity consider both satellite and the space station are static in position relative to each other for the duration of offloading. However, the offloading is not always fruitful and depends on the con-

2. www.oneweb.world
3. www.starlink.com

nection quality to the space station. If the wireless channel is used by other satellites in the vicinity, transmitting sensor data can be a lengthy process. Similarly, if the space station is already busy with other algorithmic jobs, it may take a long time to service the satellite. Both of these delays may mean that the satellite misses the threshold time for completion of the algorithm. Missing the threshold is not desirable for the satellite and should be avoided as much as possible. Additionally, the communication between the satellite and the space station may not be secure. If there is a rogue (compromised) satellite in the vicinity trying to attack either passively or actively, they may launch attacks described in the Section 2.4.

3.1 Job

A job is a computationally intensive algorithm that the satellite is trying to offload. For example, a pose estimation algorithm using camera images can be an offloading job. We define it as a tuple $< X, D, \tau_d >$ where X is the number of computation cycles per bit required to complete the job, D is the data requirement of the job, τ_d is the latest time to complete the job [21]. Such a job may have a large value for X and need significant time and energy resources to complete. In literature, jobs are considered to be offloadable or not offloadable as well as full or partial offloading [21]. However, for simplicity, we consider all the jobs to be offloadable and only full offloading is considered. The jobs are generated on a regular basis as a Poisson process with a mean arrival rate Λ.

3.2 Local Computation

A job can be processed locally using the satellite's own computing platform. In terms of security risk, as it does not involve any communication. Thus, we assume that such local computations are risk-free. However, if the device is already busy, each job has to wait for its turn. We model the local computation using Queuing Theory. We consider jobs to be processed on a First Come First Service (FCFS) without pre-emptive scheduling. The service times are dependent on the job itself. Also, when the job is offloaded with a certain security level (–see Tables 1 and 2) it will impact the processing resources depending on the security level selected. As the service times for job and security levels can be significantly different, they can be modelled as hyper-exponential distribution [46]. Lastly, the number of queue servers (N_m) is the number of cores in the device.

3.2.1 Time

Time taken by a satellite to execute a job is given by

$$\tau_m = \frac{X \times D}{I_m} \tag{1}$$

where I_m is the capability of the satellite to execute instructions usually measured in Million Instructions Per Second (MIPS). Before execution, there is a waiting time due to queuing which can be estimated using Little's law. Although, I_m may change depending on several factors including Dynamic Voltage and Frequency Scaling (DVFS) we consider a fixed policy such that the I_m does not change over the time.

3.2.2 Energy

The Central Processing Unit (CPU) power is made up of two parts, the idle power and the running power, as follows:

$$P_m = u * P_{max} + (1 - u) * P_{idle} \tag{2}$$

where, u, P_{max}, P_{idle} are the utilisation, maximum power and idle power consumption of the CPU respectively. So, energy consumed to execute a job can be calculated as

$$E_{exec} = P_m \times \tau_m. \tag{3}$$

3.3 Remote Execution

The remote platform could be the other satellites, space station or ground station. We assume the server has N_s cores available for computing, each core capable of executing I_s MIPS. Before an algorithm can be executed on the remote platform, however, the data (possibly code as well) has to be transferred to the remote platform. However, there is a risk that one of the CIA aspects is breached while communicating. So, appropriate levels of security have to be put into place. Depending on which encryption algorithm and hashing algorithm is selected (–see Tables 1 and 2), different levels of security can be maintained. Also, different algorithmic complexity of the algorithm means they will incur different times and energy costs which are described below. Similar to other algorithms in the literature, we ignore the cost of sending the result back to the device as the data is of relatively lower size in most cases.

3.3.1 Time

The total time for executing a job on a remote platform can be estimated as follows:

$$\tau_s = \tau_{security} + \tau_{comm} + \tau_w \tag{4}$$

where $\tau_{security}, \tau_{comm}, \tau_w$ represent the times taken by the *offloader* to secure, packet, send the data, and wait for receiving the result respectively. In space, the server may not be as powerful as the cloud on Earth. Hence there may be queuing as well. We model the delay in queuing with a queue similar to the queue in satellite. This is represented by τ_w. Similarly, if D is the data size to be transferred/received, the communication time is given by:

$$\tau_{comm} = \frac{D}{R} \tag{5}$$

where R is the available data rate. However, R may be shared between other users and effective bandwidth depends on other users' action. For a device k within N users sharing the communication channel, effective data rate can be calculated as

$$R_k = Bw \log_2 \left(1 + \frac{G_{k,k}.P_k}{\sigma^2 + \sum_{i=1, i \neq k}^{N} G_{i,k}.P_i} \right) \tag{6}$$

where, Bw is the bandwidth, G_i, P_i are the channel gain and transmit power for user i. As evident from Equation (6), communication data rate depends on the channel gain and transmission power, as well as other users transmitting simultaneously. Herein, 'other users' may include any devices trying to communicate on the channel. However, they could also be *jammers* trying to disrupt the communication

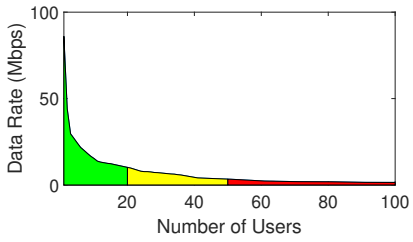

Fig. 2. Relationship between number of users and the data rate. Green region shows best network condition, yellow shows moderate and red signifies poor network condition. Availability is accounted in the model by considering poor network condition when the channel is jammed.

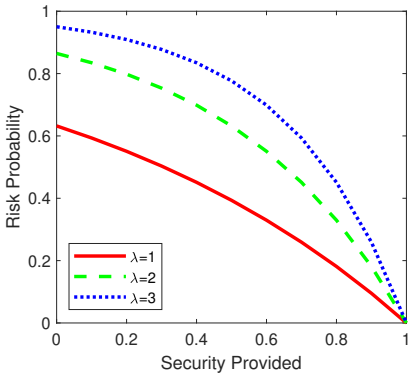

Fig. 3. Probability of security breach for SD=1

between a user and the base station. In that case, the availability is affected. We account for the availability by considering the jammed channel as poor network condition. Instead of calculating the communication rates using the transmit power and the channel gain, we use a simpler version based on arbitrary values in [47]. It only depends on the number of users actively using the channel and drops off exponentially as the number of users increases. The rates for different numbers of users are visualised in Figure 2. The actual number of satellites in the vicinity in space is random at any point in time. However, if we consider the satellites are uniformly distributed around the globe, choosing a smaller satellite formation means a lower number of satellites actively communicating on average. Similarly, if we consider a larger satellite constellation, the average number of satellites actively involved in communication may be higher.

3.3.2 Energy

The energy consumed for offloading can be calculated as:

$$E_{off} = P_c \times \tau_{comm} + P_{max} \times \tau_{security}. \tag{7}$$

3.3.3 Security Risk

The security risk increases when CO is implemented because the system is relatively more vulnerable than if the computation is done locally. Some risks can be mitigated by choosing appropriate security measures such as encryption. Similar to [45], [48] we model the risk as Poisson distribution.

$$P_k = \begin{cases} 0, & if\ SD \leq SP \\ 1 - \exp^{-\lambda^k (SD - SP)} & if\ SD > SP \end{cases} \tag{8}$$

where $k \in \{C, I\}$ is the particular security concern, SD is the security demand of the job, and SP is the chosen security level. If the security provided is greater than or equal to the security demand, then the risk is zero. However, if the chosen security is less than the required level, it is prone to security breaches. The exact probability of risk depends on λ_k which can be different for each server as well as for confidentiality and integrity. The total probability of risk such that either confidentiality or integrity is violated is then given by

$$P_r = 1 - \prod_{k \in \{C,I\}} (1 - P_k). \tag{9}$$

4 Problem Formulation

Recall that the state of the system in our problem corresponds to the number of jobs waiting in the queues at the local queue (at the satellites) and the number of jobs waiting at the server. For both these queues, the number of jobs in the next time slot will only depend on the number of jobs awaiting during the current time slot and the decisions (offloading/ local computation) taken during the current time slot. Note that for the local queues at the satellites, the number of jobs departing the task queue will depend on the decision taken during the current slot, whereas the number of new jobs arriving is assumed to be an independent and identical Poisson process (memoryless). Similarly, for the queue at the server, the number of new jobs arriving will depend on the decision taken during the current slot, whereas the departure process only depends on the computational time at the server. Note that splitting a Poisson process randomly with a fixed probability creates two separate Poisson processes. Similarly, if two Poisson processes are combined (for example two satellites may offload to the server in the same time slot) it results in a Poisson process [46]. Based on these observations, we note that, given the current state and action, the next state of the decision process is conditionally independent of all previous states and actions; in other words, the state transitions satisfy the Markov property. A Markov Decision Process (MDP) is a tuple $< S, A, T, R >$ where S is a finite set of states, A a finite set of actions, T a transition function defined as $T : S \times A \times S \rightarrow [0, 1]$ and R a reward function defined as $R : S \times T \times S \rightarrow \mathbb{R}$ [49]. We also note that our setup is similar to existing works on computational offloading [50], [51], [52], which also model the problem as a MDP and solve it using RL methods. An RL agent observes the states at discrete intervals and makes the decision for the next time interval.

Recently, there has been a growing interest in applying RL to the data and CO problem in terrestrial mobile networks. For instance, the problem of minimising the mobile user's cost, energy consumption and computation delay by offloading tasks to a mobile-edge computing server was considered in [53] and [54], and solved using Deep RL techniques. RL has been used to solve CO problem in Internet Of Thing (IOT) devices with energy harvesting as well [25], [55]. The problem of allocating computing and network resources under varying MEC conditions was considered in [56]. Reference [57] applied DRL to solve the network utility maximisation problem in a Virtualised Network Function (VNF) environment. For a detailed survey on the application

of RL in CO in wireless networks, we refer the reader to [58]. However, none of these works focus on CO in a satellite environment and the corresponding domain-specific features.

For this problem, we consider the number of jobs in the queue of the satellite, the number of jobs in the server, the number of satellites communicating in the current time slot, and the number of jobs arriving in the time slot as the observations of the system. The satellite will not always know the exact number of jobs the space station is serving. However, when the server may agree to serve the satellite, it may send regular updates on its state. In our previous work we used a proactive and reactive algorithm to send this information (Node State Information (NSI)) about one's state to neighbours [59]. The number of satellites communicating in a given time slot provides an inclination to the available bandwidth similar to the feedback channel gain. For the simulation, we assume that the number of satellites using the channel is random and independent of the previous interval but constant throughout the interval. Let P_{off} be the probability of offloading to the server. Then, the time consumption to execute a job can be estimated using Equations (1) and (4) as follows:

$$\tau = P_{off}\tau_m + (1 - P_{off})\tau_s. \tag{10}$$

Similarly, energy consumption can be estimated as

$$E = P_{off}E_m + (1 - P_{off})E_s. \tag{11}$$

While the time and energy consumption can be estimated from the system state such as the number of jobs in the CPU, communication, and the server queues, security risk cannot be observed directly or in advance. However, given enough data on previous observations and cost, we can estimate the risk conditions if it is time-invariant. The overall cost of executing is then given by

$$\begin{aligned} C_t &= \sum_{j=1}^{J}(w_t\tau_j + w_eE_j + w_rr_j) \\ subject\ to\ \tau &\leq \tau_d, \end{aligned} \tag{12}$$

where, J is the maximum number of jobs in an interval, τ_j, E_j, and r_j are time, energy and risk while executing job j. r_j is the random value sampled using Equation (9) to represent the risk. w_t, w_e, w_r are the weights for time, energy and risk components. Next, we relax the hard constraint on the time deadline to a soft constraint such that if the constraint isn't met, we add a large cost to the cost function whereas when the constraint is met the weight is zero. Also, as we propose a generic solution, we do not set these weights to custom values. Instead, we set them to equal weights. For applications that are specific, the weights could be adjusted to the application. For example, in a satellite communication network, when there are not enough satellites, the destination may not reachable and the data packets may be dropped. For such applications Delay Tolerant Network (DTN) routing protocols are used [60]. When CO used on such protocol, the weights on time can be set to zero.

$$C_t = \sum_{j=1}^{J}(w_t\tau_j + w_eE_j + w_rR_j + w_d(\tau_d - \tau)) \tag{13}$$

where,

$$w_d = \begin{cases} 0, & if\ \tau_d > \tau \\ \text{non negative number}, & if\ \tau_d \leq \tau. \end{cases} \tag{14}$$

Our objective is to minimise Equation (13) in the long term.

$$\underset{P_{off}, SL_C, SL_I}{\arg\min}\ \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t C_t\right] \tag{15}$$

where, γ is the discount factor. Our action is a three dimensional vector $[P_{off}, SL_C, SL_I]$ consisting of the probability to offload a job and the security levels to select to maintain the confidentiality and integrity of the offloaded data. Similar to [59], we select the execution platform probabilistically. The benefit of making such a decision is that the system does not need to know the exact number of incoming jobs.

4.1 Deep Deterministic Policy Gradient (DDPG)

We use DDPG [61] to solve the optimisation problem stated in Equation (15). DDPG is a actor-critic based offline method that uses two separate Deep Neural Networks (DNNs) to approximate the Q-value network. Its main advantage over DQN is its ability to work on a continuous action space [14]. So the probability to offload P_{off} can be any value ranging from $0 - 1$ and need not be discretised. We trained our reinforcement agent on episodes of simulated data with each episode lasting 40 seconds. We used the experience replay method for batch training and ADAM optimiser for the training purpose [62]. We trained the network for a maximum of 1000 episodes. To stop the agent from being greedy and making sub-optimal decisions, we use the ϵ-greedy approach whereby the agent makes a random action with a small probability ϵ and the rest of the time take the best (or greedy) action. To balance the exploration and exploitation for the agent, we gradually lowered the value of ϵ.

4.2 Deep Q-Network (DQN)

DQN [63] is the first reinforcement learning algorithm to demonstrate human level performance on Atari games. [45] used DQN based algorithm to create security aware CO algorithm. In order to compare the performance of our proposed DDPG based algorithm, we implemented a similar DQN based solution for our problem. As DQN works with a discrete action space, we quantised P_{off} with a resolution of 0.2 ranging from 0 to 1. Otherwise, the other training parameters were left same as the DDPG algorithm described below.

4.3 Static policies

We compared our proposed DDPG algorithm against three static policies defined below.

4.3.1 Local Only (LO)

As the name suggests, it cannot offload any job and is oblivious to network changes and risk states.

4.3.2 Server Only No Security (SONS)

This policy offloads all the jobs to the server without following any security guidelines. So when the risk is high, attacks are always successful.

4.3.3 Server Only Maximum Security (SOMS)

Similar to the previous policy, it offloads all the jobs. But, it uses the highest security measures regardless of the network conditions.

5 Simulation Results

In this section, we briefly explain our simulator, parameter selection and their results. We created our own simulator; all the code and environment are available at https://github.com/sausthapit/ComputationOffloadingRL. We used Matlab and Simulink environment which provides toolboxes for Reinforcement Learning (RL) and event-based simulations. The simulator also supports RL agents with discrete action spaces such as Deep Q-Network (DQN).

We assume the maximum number of satellites will be different in the three formations due to the physical setup. In particular, the trailing formation occupies the least space (as the satellites share the same orbit) and has the least number of satellites. In comparison, the swarm formation has more satellites (as it involves satellites in different orbital planes). Lastly, the constellation formation has the largest number of satellites taking part in the communication (to cover the entire earth). According to [26], the transmit power for inter-satellite communication, P_k, is in the range 0.5 W to 2 W. In the trailing formation, since the satellites are in the same orbit (and hence close to each other and less interference), we assume a lower transmit power and set $P_{comm} = 0.5$ W for the trailing formation. As the swarm and constellation formations occupy progressively larger areas of space, we assume $P_{comm} = 1$ W for the swarm formation, and $P_{comm} = 2$ W for the constellation formation. By default, we chose the swarm formation for the rest of the simulation unless specified and we set the following parameters for the simulation. The idle power (P_{idle}), execution power (P_{max}) of the satellite is set to 0.1 and 5 watts respectively, processing capability of the satellite (I_m) is set to 2.5×10^9 Million Instructions Per Second (MIPS) with $N_m = 4$. Similarly, we set the processing capability of the server satellite to be twice that of the satellite. Also, the number of cores in the server is higher than the number of cores in the satellite (i.e. $N_m < N_s = 16$). The job size is chosen to be 0.2 MB, and it takes one second to process on the satellite without the waiting times. For stability, the queues are limited to finite buffers. The maximum queue length for the Central Processing Unit (CPU) and communication buffer is set to 20 whereas, for the server, the computation buffer is set to 10. Also, τ_d is set to 5 seconds. This means if a job has to wait more than 5 seconds to process it is not useful and is considered a dropped job. Similarly, if any of the queue buffers are full when a new job arrives, it is lost as well. In default settings, we consider on average three jobs arriving per second and best network setting and lowest risk level. For the training purposes, we set the weights w_t, w_e, w_r, w_d to $1, 1, 10$ and 10 respectively. This implies whilst we would like to improve on execution times and energy, we would like to avoid losing jobs and minimise the security attack. In fact, setting w_r and w_d to the same large value suggests that a successful attack is as bad as dropping a job. However, the weight selection is done arbitrarily and can be tailored to the application requirement. For applications that are super sensitive to security threats, it could be even higher. Also, when measuring the performance instead of evaluating the actual time elapsed for each job, we count how many jobs were completed within the time threshold. Once the agent is trained, we ran 10 Monte-Carlo simulations with random seeds for each of the settings described below.

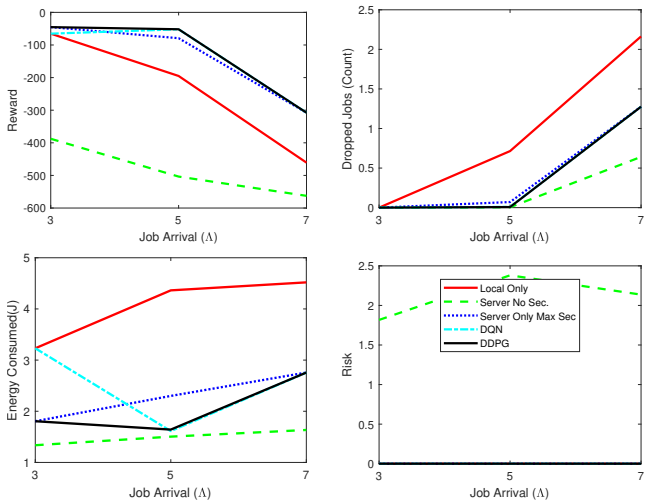

Fig. 4. Performance for different job arrival rates. Clockwise from top left: average overall score, average dropped jobs, lost jobs due to security, and energy consumed.

5.1 Incoming job rate

We simulated various job rates ranging from three jobs per second to seven jobs per second. If the satellite is equipped with a quad-core processor with a service rate of one job per second, when processing locally, its utilisation is given by

$$u = \frac{\Lambda}{N_m \mu} = \frac{4}{4 \times 1} = 1. \tag{16}$$

Hence, it is only stable for $\Lambda < 4$. Otherwise, the queue length will continue to grow indefinitely; in this case, as the buffer is limited, lost. The device is forced to offload or drop some of its jobs to maintain stability when the job rate is higher. The network condition is set to the best, risk level to the lowest, and data size to 0.2 MB. Figure 4 shows the averaged results for our experiments. The top left figure shows the overall cost achieved by each of the four policies. It is evident that the cost is growing for all the policies and Server Only No Security (SONS) fared the worst. This is due to the security attacks it has suffered the most which are seen from the bottom right image. In terms of jobs dropped (– see Figure 4, top-right), all algorithms were able to process all the jobs at $\lambda = 3$. However, as the arrival rate started to increase, the local computation suffered and dropped the most jobs averaging more than two jobs per episode. Both DQN and Deep Deterministic Policy Gradient (DDPG) have similar results with DQN using slightly more energy at lower job arrival setting.

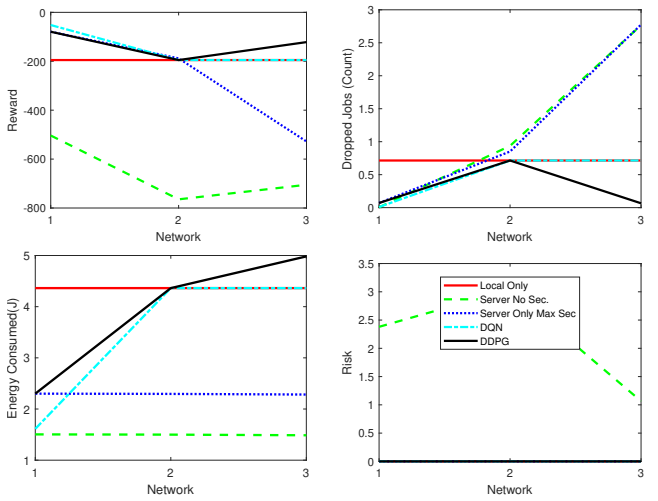

Fig. 5. Performance for different network conditions; $1, 2, 3$ represents best, medium and worse conditions. Clockwise from top left: average overall score, average dropped jobs, lost jobs due to security, and energy consumed.

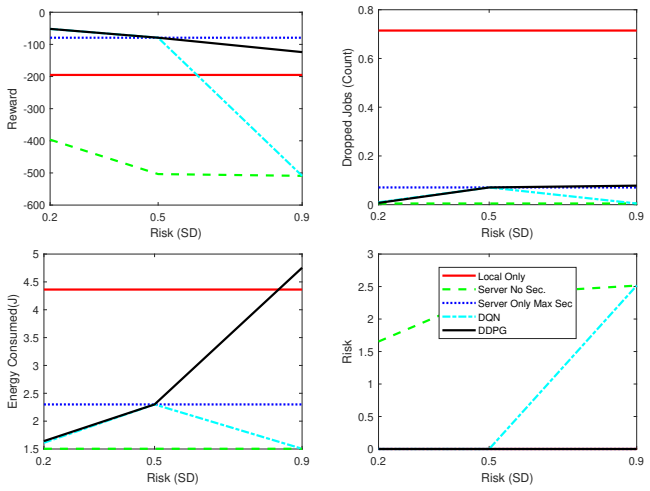

Fig. 6. Algorithm Performance for different risk conditions. Clockwise from top left: average overall score, average dropped jobs, lost jobs due to security, and energy consumed.

5.2 Network Environment

We considered three network settings, namely best, medium and poor represented by green, yellow and red area in the Figure 2. In the best setting (which is the default setting), only a few users simultaneously communicate at a given time slot, whereas in the medium setting, considerably more users communicate at the same time. Poor settings may represent a large number of satellites communicating at the same time or a malicious attacker trying to deliberately jam the channel. To simulate these settings, we simply use a uniform random number generator with boundary limits. Results for a varying network environment is presented in Figure 5. As expected the Local Only (LO) policy is not affected by the varying network condition evident by the horizontal solid red line. Both SONS and Server Only Maximum Security (SOMS) policies dropped similar amount of jobs per episode as seen in the Figure 5 top right. This is because the satellite is unable to reach the server as the network condition worsens. However, the cyan dashed line for the DDPG algorithm shows that even it performed better than the LO algorithm suggesting that it used both local and remote resources in an efficient manner. This is evident from the bottom-left figure where the cyan line is using the most energy (up to 5J at the worst network condition). In theory, the DQN should also follow a similar pattern as DDPG but in this case, when the network worsened, DQN only used the local resources.

5.3 Risk Levels

In this work, we modelled and simulated risk to emulate the real scenario of security attacks. In doing so, we changed the security desired (SD) parameter. The SD sets the security threshold that needs to be fulfilled to save from attacks. While SOMS is helpful to prevent unwanted security attacks, it uses vital computational resources, time and energy. While this may not be a problem when the resources are adequate for example on the Earth's surface with a substantial processor and mains-powered device. It can be significantly crucial to save energy and resource in space. Using remaining resources like batteries may mean the satellite or rover is completely out of service. In order to avoid this scenario, it is crucial to save as much energy as possible. Our DDPG algorithm in this instance is able to adapt to the varying security level in the environment without directly sensing it and only based on the previous results. Figure 6 shows the performance of all four policies. As usual, SONS is the only one subject to successful attacks. The DDPG algorithm used less energy than the SOMS algorithm when the security threat is pretty low (≤ 0.5) as seen in the bottom-left image. However, we also notice that when the threat is significantly high (0.9) the proposed DDPG algorithm did not offload to the server and did most of the work itself using significantly higher energy than the SOMS algorithm. DQN algorithm on the other hand was able to handle more jobs even when risk was the worst (– seen in Figure 6 top left) although more jobs were subject to security attacks. Cases such as these can be investigated further to reason why a particular agent is taking such action. One way of teaching the agent would be by changing the w_e. Furthermore, the weights could be adjusted or different agents could be combined at different environmental settings. For instance, including the remaining fuel or battery resources into the agent's observation. This way the agent can act intelligently and decide whether to prioritise security or energy resources.

5.4 Data size

The size of data has multiple effects on the performance of the simulation. As the communication time is directly proportional to the data size, doubling the data size at least doubles the transmission time. In addition, our execution time is also proportional to the data size –see Equation (1). So, the service rate of the satellite is halved when the data size is doubled. We present the results in Figure 7. We set the default data size of the algorithm to be 0.2 MB which could be the image data that the satellite is transmitting to the server for further computation. We see from the results that even under the same network and risk conditions, plenty of jobs are dropped when the data size is doubled. The LO

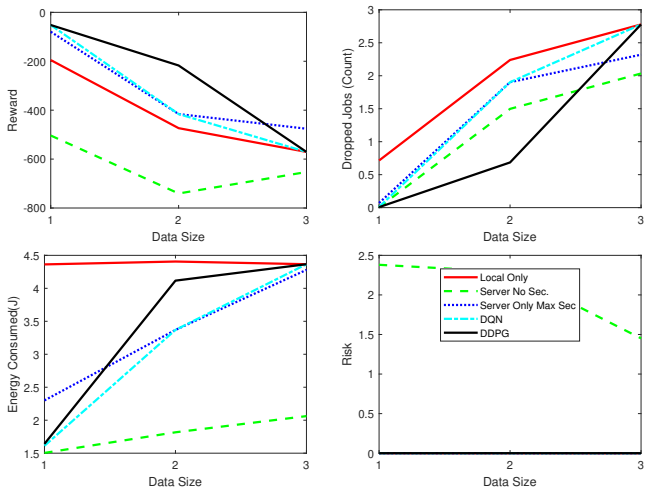

Fig. 7. Algorithm Performance for different data sizes (1, 2 or 3 times the default size). Clockwise from top left: average overall score, average dropped jobs, lost jobs due to security, and energy consumed.

policy dropped on average 2.2 jobs per episode followed closely by SOMS, then by the SONS. Our DDPG algorithm dropped the least with an average of only 0.5 jobs per episode. This improved performance came at a higher cost of energy. However, it still used less energy than the LO with had the maximum energy consumption at all data sizes. However, when the data size is tripled, DDPG algorithm dropped as many as the 2.7 jobs per episode which were the worst jointly with the local computing. In terms of overall cost, SOMS was best at the triple data size.

5.5 Satellite Formations

In Section 2.1, we described three different satellite formations namely leader/ follower, cluster and constellation. However, in previous sections, we experimented using the swarm/cluster formation only. The Round Trip Time (RTT) and the energy consumption can vary relative to the specific satellite formation [26]. We capture the dynamics of this offloading scenario in our simulation by considering different transmission energy costs and varying the number of satellites in the communication (which in turn varies the communication data rate and time delay). For the satellites in the same orbit, the number of satellites would be limited which would mean the communication delay, as well as the energy cost, is lower. Similarly, in a cluster of satellites, the number of satellites transmitting simultaneously can be higher resulting in lower data rate and higher energy consumption. Finally, in a constellation, the number of satellites communicating would be still higher due to the larger area involved.

Figure 8 and Figure 9 shows the results of the simulation for low and high incoming job rate cases respectively. In the low job rate case, none of the algorithms dropped any jobs for the trailing and cluster formation. For the constellation formation SOMS and SONS policy both dropped approximately 0.2 jobs per episode. Only SONS algorithms were subject to successful attacks as seen in the bottom right figure. However, from the bottom left figure, we see all the policies saved energy in comparison to the LO policy. But the savings decreased as the formation changed from

Formation	Power (Watt)	Maximum Satellites
Trailing	0.5	20
Cluster	1	50
Constellation	2	100

TABLE 4
Simulation parameters for different satellite formations

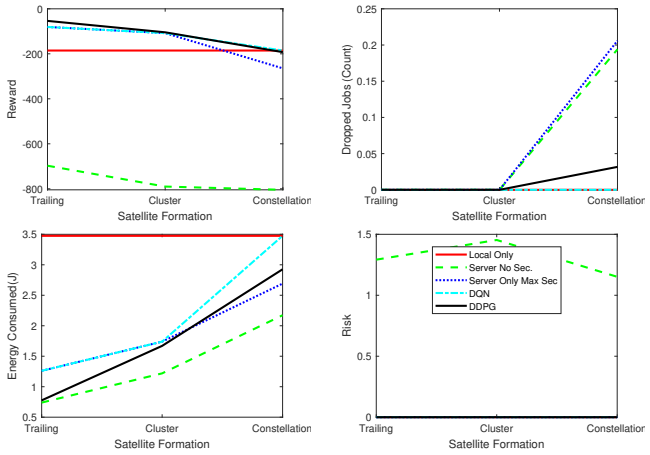

Fig. 8. Algorithm Performance for different satellite formations for low incoming job rate. Clockwise from top left: average overall score, average dropped jobs, lost jobs due to security, and energy consumed.

trailing to swarm and constellation. In the simulation setting when there were significantly more jobs present at the satellite (–see Figure 9, we see the proposed algorithm DDPG and DQN was able to save energy as well as drop fewer jobs in comparison to the LO case. The DDPG was superior to the DQN and others even in the constellation case where it dropped the least number of jobs.

From Figure 8 and Figure 9, it is also evident that DDPG is superior among all the policies. Also, it is evident that trailing formation is beneficial than local only, cluster, and the constellation formation for Computation Offloading (CO). This is because the communication channels are better than other formations. However, for all three formations, we simulated the same server capacity. As the formation grows larger in size, it may be possible to scale the server as well.

6 Conclusions and Future Work

In this work, we studied wireless communications security for space applications. In addition, we applied a useful tool for future space applications called Computation Offloading. We then solved the CO problem using a DDPG algorithm which is a robust method for solving optimisation problems in real-time. Through extensive Monte-Carlo simulations, we show that our algorithm can increase the performance of space applications. The simulations also show that the added performance comes with increased energy consumption. We compared the proposed algorithm with not only static policies but with previously published DQN method [44]. In general, the experiments showed that the DDPG is superior to DQN based method in addition to static policies. We also experimented on different satellite formations and show that the proposed algorithm is superior to the baselines. However, in some settings, such as when the risk was very high, the DQN agent performed

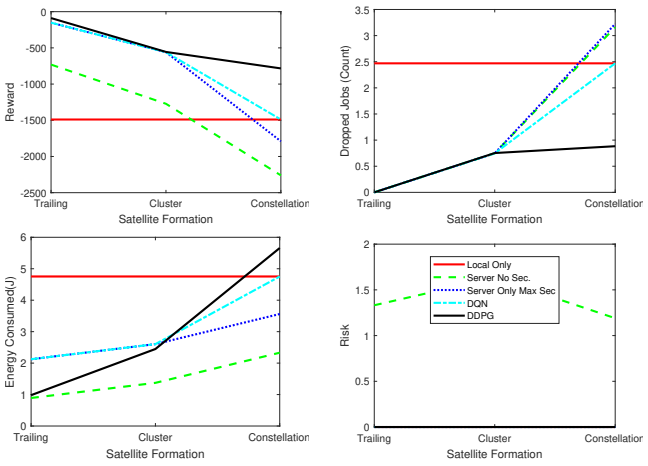

Fig. 9. Algorithm Performance for different satellite formations for high incoming job rate. Clockwise from top left: average overall score, average dropped jobs, lost jobs due to security, and energy consumed.

better. Further study is necessary to understand the cause and to understand if multiple agents can be combined to develop an even better algorithm. For *NewSpace* system where nano-satellites could work in swarms and constellations with substantial autonomy, it is vital that satellites and rovers can trust each other and depend on each other. The algorithm presented in this paper can find applications in this new environment. In the future, we would like to incorporate further contexts such as the remaining energy resource of the satellites as well as the movement of satellites and authentication issues.

Acknowledgments

This work is supported by grant EP/R026092 (FAIR-SPACE Hub) through UKRI under the Industry Strategic Challenge Fund (ISCF) for Robotics and AI Hubs in Extreme and Hazardous Environments

References

[1] Xilinx, "RT Kintex UltraScale FPGAs for Ultra High Throughput and High Bandwidth Applications," May 2020.

[2] D. Lüdtke, K. Westerdorff, K. Stohlmann, A. Börner, O. Maibaum, T. Peng, B. Weps, G. Fey, and A. Gerndt, "OBC-NG: Towards a reconfigurable on-board computing architecture for spacecraft," in *2014 IEEE Aerospace Conference*, Mar. 2014, pp. 1–13.

[3] M. Pignol, "COTS-based applications in space avionics," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, Mar. 2010, pp. 1213–1219.

[4] J. Son, T. He, and R. Buyya, "CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments," *Software: Practice and Experience*, vol. 49, no. 12, pp. 1748–1764, Dec. 2019.

[5] S. Xu, X. Wang, and M. Huang, "Software-Defined Next-Generation Satellite Networks: Architecture, Challenges, and Solutions," *IEEE Access*, vol. 6, pp. 4027–4041, 2018.

[6] Gary Martin, "NewSpace: The Emerging Commercial Space Industry," https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160001188.pdf.

[7] K. Woellert, P. Ehrenfreund, A. J. Ricco, and H. Hertzfeld, "Cubesats: Cost-effective science and technology platforms for emerging and developing nations," *Advances in Space Research*, vol. 47, no. 4, pp. 663–684, Feb. 2011.

[8] B. Denby and B. Lucia, "Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 939–954.

[9] I. F. Akyildiz and A. Kak, "The Internet of Space Things/CubeSats," *IEEE Network*, vol. 33, no. 5, pp. 212–218, Sep. 2019.

[10] C. Maple, M. Bradbury, H. Yuan, M. Farrell, C. Dixon, M. Fisher, and U. I. Atmaca, "Security-Minded Verification of Space Systems," in *2020 IEEE Aerospace Conference*, Mar. 2020, pp. 1–13.

[11] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci, "CloudScout: A Deep Neural Network for On-Board Cloud Detection on Hyperspectral Images," *Remote Sensing*, vol. 12, no. 14, p. 2205, Jan. 2020.

[12] G. Furano, G. Meoni, A. Dunne, D. Moloney, V. Ferlet-Cavrois, A. Tavoularis, J. Byrne, L. Buckley, M. Psarakis, K.-O. Voss, and L. Fanucci, "Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44–56, Dec. 2020.

[13] UK Space Agency, "Take-off for UK-built supercomputer nanosatellites," https://www.gov.uk/government/news/take-off-for-uk-built-supercomputer-nanosatellites.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Proc. ICLR*, 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[15] S. Sadr, A. Anpalagan, and K. Raahemifar, "Radio resource allocation algorithms for the downlink of multiuser ofdm communication systems," *IEEE Communications Surveys Tutorials*, vol. 11, no. 3, pp. 92–106, 2009.

[16] L. Song, D. Niyato, Z. Han, and E. Hossain, "Game-theoretic resource allocation methods for device-to-device communication," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 136–144, 2014.

[17] S. Lakshminarayana, M. Assaad, and M. Debbah, "H-infinity control based scheduler for the deployment of small cell networks," in *Proc. IEEE International Symposium of Modeling and Optimization of Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2011, pp. 9–16.

[18] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, 2016.

[19] S. Lakshminarayana, J. S. Karachiwala, S.-Y. Chang, G. Revadigar, S. L. S. Kumar, D. K. Yau, and Y.-C. Hu, "Signal jamming attacks against communication-based train control: Attack impact and countermeasure," in *Proc. ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, 2018, p. 160–171.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[21] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 24.

[22] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[23] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 23.

[24] S. Sthapit, J. R. Hopgood, and J. Thompson, "Distributed computational load balancing for real-time applications," in *2017 25th European Signal Processing Conference (EUSIPCO)*. Kos, Greece: IEEE, Aug. 2017, pp. 1385–1189.

[25] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-Based Computation Offloading for IoT Devices With Energy Harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[26] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh, "Survey of Inter-Satellite Communication for Small Satellite Systems: Physical Layer to Network Layer View," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2442–2473, Fourthquarter 2016.

[27] Z. Qu, G. Zhang, H. Cao, and J. Xie, "LEO Satellite Constellation for Internet of Things," *IEEE Access*, vol. 5, pp. 18 391–18 401, 2017.

[28] H. Nishiyama, D. Kudoh, N. Kato, and N. Kadowaki, "Load Balancing and QoS Provisioning Based on Congestion Prediction for GEO/LEO Hybrid Satellite Networks," *Proceedings of the IEEE*, vol. 99, no. 11, pp. 1998–2007, Nov. 2011.

[29] F. Davoli, C. Kourogiorgas, M. Marchese, A. Panagopoulos, and F. Patrone, "Small satellites and CubeSats: Survey of structures, architectures, and protocols," *International Journal of Satellite Communications and Networking*, vol. 37, no. 4, pp. 343–359, 2019.

[30] A. Kak and I. F. Akyildiz, "Large-Scale Constellation Design for the Internet of Space Things/CubeSats," in *2019 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[31] J. Mukherjee and B. Ramamurthy, "Communication Technologies and Architectures for Space Network and Interplanetary Internet," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 881–897, Second 2013.

[32] I. F. Akyildiz, E. Ekici, and M. D. Bender, "MLSR: A novel routing algorithm for multilayered satellite IP networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 411–424, Jun. 2002.

[33] N. Saeed, A. Elzanaty, H. Almorad, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, "CubeSat Communications: Recent Advances and Future Challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1839–1862, thirdquarter 2020.

[34] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, J. Querol, L. Lei, T. X. Vu, and G. Goussetis, "Satellite Communications in the New Space Era: A Survey and Future Challenges," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 70–109, 2021.

[35] Z. Song, Y. Hao, Y. Liu, and X. Sun, "Energy Efficient Multi-Access Edge Computing for Terrestrial-Satellite Internet of Things," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[36] Z. Qu, G. Zhang, H. Cao, and J. Xie, "LEO Satellite Constellation for Internet of Things," *IEEE Access*, vol. 5, pp. 18 391–18 401, 2017.

[37] S. Yu, X. Gong, Q. Shi, X. Wang, and X. Chen, "EC-SAGINs: Edge Computing-enhanced Space-Air-Ground Integrated Networks for Internet of Vehicles," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[38] Todd Harrison, Kaitlyn Johnson, Thomas G. Roberts, Tyler Way, and Makena Young, "Space Threat Assessment 2020," Center for Strategic and International Studies, Tech. Rep., 2020.

[39] M. Bradbury, C. Maple, H. Yuan, U. I. Atmaca, and S. Cannizzaro, "Identifying Attack Surfaces in the Evolving Space Industry Using Reference Architectures," in *2020 IEEE Aerospace Conference*, Mar. 2020, pp. 1–20.

[40] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 6th ed. USA: Prentice Hall Press, 2013.

[41] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, Jun. 2017.

[42] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *IEEE Transactions on Computers*, vol. 55, no. 7, pp. 864–879, Jul. 2006.

[43] H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, "Scheduling for Workflows with Security-Sensitive Intermediate Data by Selective Tasks Duplication in Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 9, pp. 2674–2688, Sep. 2017.

[44] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *arXiv:1808.01977 [cs]*, Aug. 2019.

[45] B. Huang, Y. Li, Z. Li, L. Pan, S. Wang, Y. Xu, and H. Hu, "Security and Cost-Aware Computation Offloading via Deep Reinforcement Learning in Mobile Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–20, Dec. 2019.

[46] W. J. Stewart, *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton, N.J: Princeton University Press, 2009.

[47] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[48] T. Xiaoyong, K. Li, Z. Zeng, and B. Veeravalli, "A Novel Security-Driven Scheduling Algorithm for Precedence-Constrained Tasks in Heterogeneous Distributed Systems," *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 1017–1029, Jul. 2011.

[49] M. van Otterlo and M. Wiering, "Reinforcement learning and markov decision processes," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42.

[50] T. Truong-Huu, C.-K. Tham, and D. Niyato, "To Offload or to Wait: An Opportunistic Offloading Algorithm for Parallel Tasks in a Mobile Cloud," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. Singapore, Singapore: IEEE, Dec. 2014, pp. 182–189.

[51] Y. Zhang, D. Niyato, and P. Wang, "Offloading in Mobile Cloudlet Systems with Intermittent Connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.

[52] M. Abu Alsheikh, D. T. Hoang, D. Niyato, H.-P. Tan, and S. Lin, "Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1239–1267, 23.

[53] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[54] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[55] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[56] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2019.

[57] L. Gu, D. Zeng, W. Li, S. Guo, A. Y. Zomaya, and H. Jin, "Intelligent vnf orchestration and flow scheduling via model-assisted deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 279–291, 2020.

[58] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[59] S. Sthapit, J. Thompson, N. M. Robertson, and J. R. Hopgood, "Computational Load Balancing on the Edge in Absence of Cloud and Fog," *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1499–1512, Jul. 2019.

[60] M. A. A. Madni, S. Iranmanesh, and R. Raad, "DTN and Non-DTN Routing Protocols for Inter-CubeSat Communications: A comprehensive survey," *Electronics*, vol. 9, no. 3, p. 482, Mar. 2020.

[61] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, Jul. 2019.

[62] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017.

[63] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

Saurav Sthapit is a Research Fellow in Cyber Systems Engineering in WMG at the University of Warwick. He received the BE degree in electronics and communication engineering from Tribhuvan University, Nepal, the MSc degree in embedded systems from the University of Kent, England and the PhD degree in the Institute for Digital Communications, within the School of Engineering, University of Edinburgh, Scotland. His research interests include computer vision, mobile computing, cyber security, and reinforcement learning, etc.

Subhash Lakshminarayana (S'07, M'12, SM'20) is an assistant professor at the School of Engineering, University of Warwick, UK. Previously, he worked as a researcher at the Advanced Digital Sciences Center (ADSC) in Singapore between 2015-2018, a joint post-doctoral researcher at Princeton University and the Singapore University of Technology and Design (SUTD) between 2013-2015. He received his Ph.D. from the Alcatel Lucent Chair on Flexible Radio and the Department of Telecommunications at SUPELEC, France in 2013, M.S. degree in Electrical and Computer Engineering from The Ohio State University in 2009 and B.S. from Bangalore University, India.

His research interests include cyber-physical system security (power grids and urban transportation) and wireless communications. His works have been selected among the Best conference papers on integration of renewable & intermittent resources at the IEEE PESGM - 2015 conference, and the "Best 50 papers" of IEEE Globecom 2014 conference. He serves as Associate Editor at the IET smart grid journal and Frontiers in Communications and Networks Journal (Smart Grid Communications section), and regularly serves in the technical program committees of IEEE conferences.

Ligang He (Member, IEEE) is currently a Reader with the Department of Computer, The University of Warwick. He has published more than 130 articles in international conferences and journals, such as the IEEE TC, TPDS, TACO, IPDPS, SC, and VLDB. His research interests include parallel and distributed processing and big data processing.

Gregory Epiphaniou currently holds a position as an Associate Professor of security engineering at the University of Warwick. His role involves bid support, applied research and publications. Part of his current research activities is formalised around cyber effects modeling, wireless communications with the main focus on crypto-key generation, exploiting the time-domain physical attributes of V-V channels and cyber resilience. He led and contributed to several research projects funded by EPSRC, IUK and local authorities totalling over £4M. He currently holds a subject matter expert panel position in the Chartered Institute for Securities and Investments. He acts as a technical committee member for several scientific conferences in Information and network security and served as a key member in the development of WS5 for the formation of the UK Cybersecurity Council.

Carsten Maple is the Principal Investigator of the NCSC-EPSRC Academic Centre of Excellence in Cyber Security Research, and Professor of Cyber Systems Engineering in WMG, at the University of Warwick. He is also a co-investigator of the PETRAS National Centre of Excellence for IoT Systems Cybersecurity where he leads on Transport & Mobility. He is a Fellow of the Alan Turing Institute, the National Institute for Data Science and AI in the UK. He leads a large project portfolio, receiving funding from a range of national and international organisations. His research interests lie in developing methods for security, privacy and resilience.