Applied Computing Review

Jun. 2018, Vol. 18, No. 2

Frontmatter

Editors		**3**
SIGAPP FY'18 Quarterly Report	J. Hong	**4**

Selected Research Articles

An Attack Scenario and Mitigation Mechanism for Enterprise BYOD Environments	S. Siboni, A. Shabtai, and Y. Elovici	**5**
Selecting Suitable Virtual Machine Migrations for Optimal Provisioning of Virtual Data Centers	M. Gilesh, S. Satheesh, S. Kumar, and L. Jacob	**22**
Combining Deep Reinforcement Learning with Prior Knowledge and Reasoning	N. Bougie, L. Cheng, and R. Ichise	**33**
An Extended Flexible Job Shop Scheduling Problem with Parallel Operations	W. Lunardi and H. Voos	**46**

Applied Computing Review

Editor in Chief: Sung Y. Shin

Associate Editors: Hisham Haddad, Jiman Hong, John Kim, Tei-Wei Kuo, Maria Lencastre

Editorial Board Members

Gail-Joon Ahn
Rachid Anane
Davide Ancona
Joao Araujo
Massimo Bartoletti
Thais Batista
Maurice Ter Beek
Giampaolo Bella
Albert Bifet
Stefano Bistarelli
Gloria Bordogna
Marco Brambilla
Barrett Bryant
Andre Carvalho
Jaelson Castro
Tomas Cerny
Priya Chandran
Li-Pin Chang
Li Chen
Seong-je Cho
Soon Ae Chun
Juan Manuel Corchado
Luís Cruz-Filipe
Marilia Curado
Lieven Desmet
Marco Di Natale
Mauro Dragoni
Khalil Drira
Mohammed El Hassouni
Yliès Falcone
Mario Freire
Joao Gama
Karl Goeschka
George Hamer
Hyoil Han
Jingtong Hu
Yin-Fu Huang
Jun Huang
Hasan Jamil
Rachid Jennane
Jinman Jung
Eric Kergosien
Bongjae Kim
Dongkyun Kim
Sang-Wook Kim
SD Madhu Kumar
Paola Lecca
Byungjeong Lee
Jaeheung Lee
Axel Legay
Hong Va Leong
Frederic Loulergue
Tim A. Majchrzak
Leonardo Mariani
Cristian Mateos
Rosa Meo
Marjan Mernik
Armin Mikler
Marina Mongiello
Eduardo Mosqueira-Rey
Flavio Oquendo
Apostolos Papadopoulos
Gabriella Pasi
Anand Paul
Ronald Petrlic
Francesco Poggi
Rajeev Raje
Rui P. Rocha
Agostinho Rosa
Marco Rospocher
Davide Rossi
Gwen Salaun
Francesco Santini
Patrizia Scandurra
Abdelhak-Djamel Seriai
Eunjee Song
Christoph Sorge
Junping Sun
Sangsoo Sung
Jean-Pierre Talpin
Maguelonne Teisseire
Dan Tulpan
Wei Wang
Jason Xue
Markus Zanker
Tao Zhang
Yong Zheng
Jun Zheng
Tewfik Ziadi

SIGAPP FY'18 Quarterly Report

April 2018 – June 2018
Jiman Hong

Mission

To further the interests of the computing professionals engaged in the development of new computing applications and to transfer the capabilities of computing technology to new problem domains.

Officers

Chair	Jiman Hong Soongsil University, South Korea
Vice Chair	Tei-Wei Kuo National Taiwan University, Taiwan
Secretary	Maria Lencastre University of Pernambuco, Brazil
Treasurer	John Kim Utica College, USA
Webmaster	Hisham Haddad Kennesaw State University, USA
Program Coordinator	Irene Frawley ACM HQ, USA

Notice to Contributing Authors

By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to include the article in the ACM Digital Library and in any Digital Library related services
- to allow users to make a personal copy of the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will refer requests for republication directly to you.

SAC 2018

The 2018 Symposium on Applied Computing was held in Pau, France, and the information about award winners can be found at https://www.sigapp.org/sac/sac2018/awards.html.

Next Issue

The planned release for the next issue of ACR is September 2018.

An Attack Scenario and Mitigation Mechanism for Enterprise BYOD Environments

Shachar Siboni
Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
sibonish@post.bgu.ac.il

Asaf Shabtai
Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
shabtaia@bgu.ac.il

Yuval Elovici
Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
elovici@inter.net.il

ABSTRACT
The recent proliferation of the Internet of Things (IoT) technology poses major security and privacy concerns. Specifically, the use of personal IoT devices, such as tablets, smartphones, and even smartwatches, as part of the Bring Your Own Device (BYOD) trend, may result in severe network security breaches in enterprise environments. Such devices increase the attack surface by weakening the digital perimeter of the enterprise network and opening new points of entry for malicious activities. In this paper we demonstrate a novel attack scenario in an enterprise environment by exploiting the smartwatch device of an innocent employee. Using a malicious application running on a suitable smartwatch, the device imitates a real Wi-Fi direct printer service in the network. Using this attack scenario, we illustrate how an advanced attacker located outside of the organization can leak/steal sensitive information from the organization by utilizing the compromised smartwatch as a means of attack. An attack mitigation process and countermeasures are suggested in order to limit the capability of the remote attacker to execute the attack on the network, thus minimizing the data leakage by the smartwatch.

CCS Concepts
• Security and privacy → Network security; Security in hardware → Embedded systems security; Systems security.

Keywords
Internet of Things (IoT), Bring Your Own Device (BYOD), Security, Rogue Access Point, Data Leakage, Smartwatch, Wi-Fi Direct Printers, Enterprise Networks, Attack Mitigation, Countermeasures.

1. INTRODUCTION
The Internet of Things (IoT) defines a new era where ordinary devices, including home appliances, medical equipment, organizational and industrial infrastructure, wearable devices, and more, are transformed into smart connected devices with the ability to sense, compute, and communicate with their surroundings via the Internet [1, 2]. However, the proliferation of IoT technology and its applications poses major security and privacy risks, especially due to their unobtrusive nature and wide range of capabilities [3, 4]. IoT devices are typically constrained by limited resources in terms of power source, memory size, bandwidth communication, and computational capabilities; this may results in security breaches, particularly due to the security mechanisms capable of being implemented in the devices (e.g., lightweight authentication and encryption algorithms). In addition, IoT manufactures are not patching old devices and focus only on improving new ones. Consequently, IoT devices can be easily compromised [5]. Moreover, as they routinely transmit, receive, and process sensitive data, including user, network traffic, and physical sensor information (e.g., transportation and energy control), these devices have become an attractive target for hackers and are exposed to the newest security breaches [6]. Furthermore, the integration of IoT devices in existing networks and system environments increases the attack surface of extant networks and systems by becoming new points of entry and leakage [7]. Specifically, the use of mobile IoT devices, such as tablets, smartphones, and smartwatches in organizational environments as part of the Bring Your Own Device (BYOD) trend, may result in severe security and privacy issues [8, 9].

In the Enterprise Internet of Things (EIoT) domain, most IoT solutions for enterprises are targeted at increasing productivity, automating human processes, and reducing costs [10]. Nevertheless, the cost of deploying IoT solutions in organizational environments, together with the impact on data privacy and security, raise major concerns [11, 12]. While IT departments can control and manage standard computer devices, such as servers, PCs, laptops, and more, most companies are not yet aware of the issues associated with the IoT or prepared to secure IoT deployments on their premises. Moreover, the tradeoff between cost, usability, and security/privacy issues is more severe in an EIoT setting [13]. For example, compromised IoT devices deployed in enterprise environments can lead to dangerous scenarios including: industrial espionage by attacking video conference and IP-based telecommunication systems; destroying critical equipment by compromising climate control in HVAC systems; disabling physical entrances to offices; and leaking private company and user information by extracting Wi-Fi and user credentials [7]. As a result, sensitive corporate and private information might become more accessible to attackers and can be

Copyright is held by the authors. This work is based on an earlier work: SAC'18 Proceedings of the 2018 ACM Symposium on Applied Computing, Copyright 2018 ACM 978-1-4503-5191-1. http://dx.doi.org/10.1145/3167132.3167214.

exposed to unauthorized individuals via these smart connected devices [3, 14].

Cyber-attacks are often based on remote connectivity with the attack target [15]. In enterprise environments, several types of security mechanisms for intrusion prevention and detection are typically employed in order to block unauthorized connections. Therefore, the first step of a remote attack is typically to inject a spearhead malware into the enterprise network by exploiting a variety of means, such as a forged link, an infected attachment, and more. Using this approach, a remote attacker may gain access to valuable assets in the network and unleash a major advanced persistent threat attack [16]. Such attacks require that the external attackers have special skills, are considered quite complex to realize, and often demand ongoing monitoring of the attack target. Another possible attack scenario in enterprise environments is posed by inside attackers such as disgruntled employees who might attempt to gain access to private and classified information by any means [17]. In both cases, legitimate IoT devices can be utilized in order to disguise the attack [18].

In this paper we demonstrate a novel attack scenario in an enterprise environment by utilizing the smartwatch device of an employee. First, the smartwatch, which was infected with a malware in advance (e.g., as a result of a supply chain attack or by downloading a legitimate application that covertly acts as a malicious application on demand during synchronization with the paired smartphone device), gathers information about the available Wi-Fi direct printers that exist in the network. Then, the malicious application opens a rogue access point with the same SSID (service set identifier) name as the selected (Wi-Fi direct) printer, thereby using the compromised smartwatch device to imitate a real printing service in the network, resulting in a fake Wi-Fi direct printer in the network. In so doing, whenever an enterprise user/employee that is close to the compromised smartwatch sends a print job to the Wi-Fi direct printer, the print job is actually processed by the compromised smartwatch device instead of the real printer. Using this scenario we illustrate how an advanced attacker located outside of the organization can exploit the compromised smartwatch device of an innocent employee to intercept print jobs sent to a legitimate Wi-Fi direct printer deployed in the network in order to leak/steal sensitive data from the organization.

To mitigate and prevent this attack scenario from being executing by the compromised smartwatch device in the organization, we proposed an attack mitigation process and countermeasures as follows. First, a preliminary security acceptance testing procedure is conducted by the sysadmin/security officer of the organization in order to assess the possible security risks of the smartwatch device to the organization; this testing procedure is performed before allowing the use of the smartwatch device in the network, and includes a device feature and capability test, a known vulnerability assessment test, and a user applications analysis test. Next, real-time network traffic monitoring and peripheral scanning of the wireless Wi-Fi access points deployed in the network are employed in order to detect abnormal behavior of the smartwatch in the network. Finally, based on alerts and notifications obtained from the monitoring, countermeasures are proactively performed, including device quarantine and isolation, and rogue access point deactivation, in order to limit the capability of the remote attacker from executing the attacks in the network by utilizing the smartwatch device as means of attack.

The contribution of the paper is as follows:

1. We raise concern about using Internet of Things (IoT) devices in enterprise environments, and specifically mobile IoT devices as part of the Bring Your Own Device (BYOD) trend.

2. We define a novel attack model that illustrates how to utilize mobile IoT devices, and specifically smartwatch devices, in enterprise networks for advanced attacks.

3. We implement a practical proof of concept showing how a capable smartwatch device can be exploited as a rogue access point that imitates a real Wi-Fi printer service and is used for leaking data sent to wireless enterprise printers.

4. We evaluate the feasibility of the attack scenario by conducting connectivity tests (based on the signal strength and distance from the real and rogue access points), and performance tests (such as the number of simultaneous connections enabled and maximum file size that can be processed by the fake Wi-Fi printer service), proving the feasibility of the proposed attack model.

5. We propose an attack mitigation process for the presented attack scenario by employing several countermeasure techniques, including a preliminary security acceptance testing procedure, real-time network traffic monitoring, and proactively performed prevention operations, which are used to prevent the compromised smartwatch device from executing the attack in the organization.

The rest of the paper is structured as follows. In Section 2, we portray the attack model for the scenario discussed, followed by an in depth description of the attack scenario evaluation process in Section 3. In Section 4 we describe an attack mitigation process and countermeasures for the attack scenario. In Section 5 we present work related to this study, and we conclude and suggest possible future work in Section 6.

2. ATTACK MODEL

The attack model considered in this paper involves a member of the organization (an employee) that uses a smartwatch device that is compromised by a remote attacker (located outside of the organization) in order to steal confidential information from the enterprise network of the organization. The attack is focused on leaking confidential information that is sent to a Wi-Fi direct printer deployed in the enterprise network. To achieve this, the attacker compromises the employee's smartwatch at his/her home possible when the employee connects the smartwatch to the smartphone device for syncing with a cloud service. Accordingly, the compromised smartwatch device is infected with a malicious application that imitates a real Wi-Fi direct printer service in the enterprise network of the organization (where the victim user works) by opening a rogue access point with an SSID name identical to a real Wi-Fi direct printer deployed in the network. As a result, the compromised smartwatch device runs a fake Wi-Fi direct printer in the network.

The attack is comprised of several phases, including a reconnaissance phase, in which the compromised smartwatch gathers information about the Wi-Fi printers deployed in the network using a malicious application; a traffic analysis phase, in

which the attacker analyzes and reconstructs (offline) the protocol messages transferred between a legitimate computer/laptop and a (targeted) Wi-Fi direct printer in order to accurately imitate a real Wi-Fi printer service during the attack; and a leakage phase, in which the attack is executed using the (same) malicious application (running on the compromised smartwatch) that in this phase imitates a real Wi-Fi direct printing service. The following is a detailed description of the attack model's phases.

2.1 Reconnaissance Phase

In the reconnaissance phase, the attacker gathers as much information as possible about the enterprise network, and more specifically, the Wi-Fi printers deployed in the network, using a compromised smartwatch device. This can be done by utilizing a malicious application [19] that runs on the smartwatch and performs a reconnaissance process in the network.

The malicious application can perform an active reconnaissance process by utilizing an Nmap security scanner tool [20] that has been modified to run in an Android Wear environment [19], as demonstrated in Figure 1. In this case, Nmap is used as an attack tool that requires only a standard mode of operation, without the need for a rooted device, in order to map the entire IP-based enterprise network. The information received during the reconnaissance phase includes all IP addresses, services, and TCP/IP ports each client or server deployed in the network is running. Specifically, for network printers, two main services are typically included: the raw TCP/IP printing service identified by TCP port 9100 and the Internet Printing Protocol (IPP) identified by TCP/UDP port 631 (illustrated in Figure 2). Another option is to map all Wi-Fi direct printers that are open and deployed in the network using a passive reconnaissance process by utilizing the 'airodump-ng' tool (as part of the 'aircrack-ng' package [21]). This option reveals the encryption mode of the printers, both password protected mode (WPA/WPA2) and unprotected mode (OPN) where no password is defined. This can be used to target the attack on open Wi-Fi direct printers that are not password protected, which is the case in this paper.

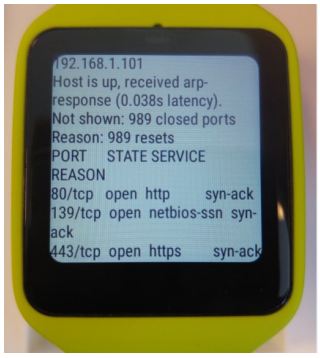


Figure 1: Active reconnaissance process executed by a smartwatch device using the Nmap security scanner tool.

All of the network mapping information gathered in this phase (including the network printers) is stored locally in the smartwatch device, and later sent to a cloud service of the remote attacker (possibly when the smartwatch is synchronized with its paired smartphone device). Based on the information obtained, the target printer for the attack is selected.

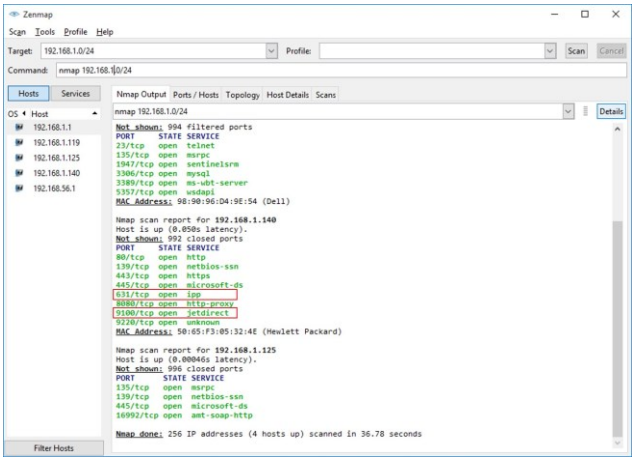


Figure 2: Printing services and TCP/IP ports of network printers identified by the Nmap tool.

2.2 Traffic Analysis Phase

In the traffic analysis phase, the attacker analyzes (offline) the protocol messages transferred between his/her computer/laptop and the Wi-Fi network printer selected during the previous reconnaissance phase. The attacker does this by deploying the same Wi-Fi printer model in his/her lab in order to reconstruct the specific printing process of the targeted printer. This information is then used in order to run a fake Wi-Fi printer service on the smartwatch device by the malicious application.

Accordingly, the attacker employs the Wireshark network protocol analyzer tool [22]. Wireshark enables the attacker to monitor and record the network traffic observed during various stages of legitimate printing processes, including: installation of the printer driver on the client (computer/laptop) and idle mode, where only control messages are exchanged between the client and the printer; and during the submission and execution of a print job. The most important information gathered in the traffic analysis phase includes: printing protocol control messages exchanged during the printing process, such as IPP control messages via TCP port 631 (shown in Figure 3); printer attribute messages that convey functional information regarding the internal status of the printer (e.g., ink status); status indication of the print job (i.e., printer spooling queue status); and the use of PCL (Printer Command Language) as the data format of information exchanged between the computer/laptop and the printer during the printing process. Again, this information is then used to simulate a fake Wi-Fi printer service running on the compromised smartwatch device.

2.3 Leakage Phase

In the leakage phase, a malicious application running on a compromised smartwatch device [19] (also shown on smartphone in [23]) opens a rogue access point (AP) with the same SSID name of a legitimate Wi-Fi direct printer (identified and selected by the attacker in advance in the previous phases) and imitates a real printing service on the network. The malicious application can be any legitimate application for smartwatch devices that covertly acts as a malicious application on demand, illustrated in Figure 4. For example, a dummy application runs as a clock

(showing the time and date, as illustrated in Figure 4.a), but in the background it acts as a real printer service and processes all print jobs (received in PCL data format) that are sent to the compromised smartwatch device (as depicted in Figure 4.b).

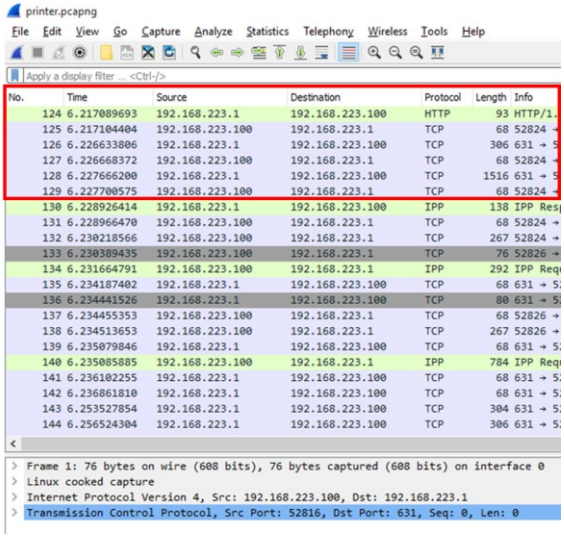
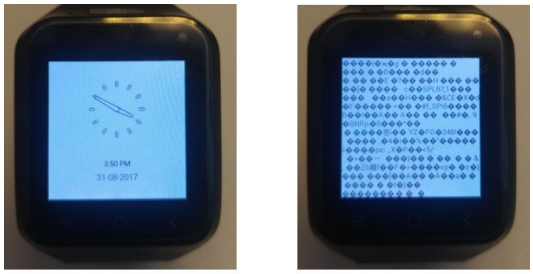


Figure 3: Exchange of printing protocol messages between a legitimate computer/laptop and a Wi-Fi direct printer.



(a) (b)

Figure 4: Malicious application running on the smartwatch device: (a) MyClock Dummy application; (b) Processing a print job (in PCL data format) using the smartwatch device.

Using our attack model, the smartwatch device poses as a legitimate Wi-Fi direct printer in the network in order to unobtrusively collect sensitive information from the organization. Once a print job has been processed by the smartwatch device, a notification is sent to the attacker, informing he/she about the print job that was recently intercepted by the smartwatch. The processed print job is stored locally on the smartwatch device and is later sent to a cloud service of the remote attacker (possibly during synchronization with the paired smartphone device).

Two important issues need to be considered in the leakage phase. First, the need to deactivate the real Wi-Fi direct printer deployed on the network before the attack is executed, so that the upcoming print jobs will be processed by the compromised smartwatch instead of the real printer (this can be done by deauthenticating connected clients and re-authenticating them with the smartwatch using the 'aireplay-ng' tool as part of the 'aircrack-ng' package [21]). Second, in case of password protected printers, there is a need to crack the password set by the user/administrator.

3. ATTACK SCENARIO EVALUATION

The setup configuration for the attack scenario is depicted in Figure 5, and comprises an HP OfficeJet 8610 Wi-Fi direct enabled printer, a ZGPAX S8 smartwatch/phone device (Android 4.4.3 OS, CPU MTK6572 Dual Core 1GHz, ROM 4GB, WI-FI 802.11b/g/n, Bluetooth, GPS and 3G GSM+WCDMA), and Dell laptop serving as the workstation of the (victim) user/employee.

3.1 Attack Scenario Process

The attack scenario, illustrated in Figure 6, includes the following phases. The victim user wears a compromised smartwatch device while using his/her laptop, as shown in Figure 5. The smartwatch, which runs our malicious application [19], imitates a legitimate Wi-Fi direct printer service in the network. The smartwatch device, a ZGPAX S8 smartwatch/phone device, runs a rogue access point with the same SSID name of the HP OfficeJet 8610 Wi-Fi direct printer (specifically 'HP-Print-B2-Officejet Pro 8610' as the factory setting of the printer).

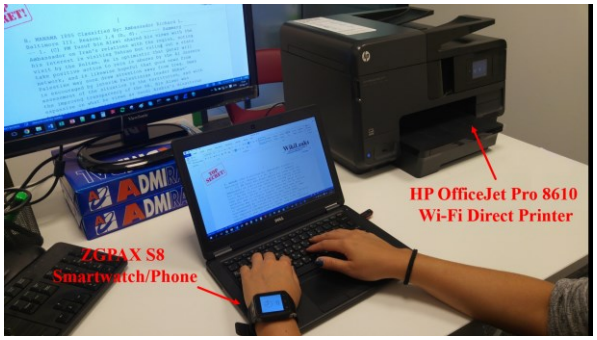


Figure 5: Setup configuration for the attack scenario, showing the targeted printer and the victim user wearing a compromised smartwatch device while using the laptop.

Later, when the victim user wants to send a print job from the laptop, he/she selects the printer based on its SSID name, inadvertently connecting the laptop to the fake Wi-Fi direct printer instead of the real printer, and sends the print job (Phase 1 in Figure 6). The compromised smartwatch then receives the print job (sent by the victim user) in PCL data format. To reduce suspicion, a local process is performed on the smartwatch (Phase 2 in Figure 6) to send the user a status notification about the print job's progress, as illustrated in Figure 7.

Once the ZGPAX S8 smartwatch/phone device finishes processing the print job, the PCL file (.txt file) is stored locally on the smartwatch device; it is later sent to a cloud service of the remote attacker when an Internet connection is established, possibly via the smartwatch's 3G cellular network connectivity, or when the smartwatch is synchronized with its paired smartphone device (Phase 3 in Figure 6). In order to cover the attacker's tracks, the malicious application running on the compromised smartwatch closes the rogue AP; therefore, if the victim user realizes that the print job isn't printed by the printer and sends the print job to the printer again (at least one more time as is typical user behavior in such situations), the print job will be processed by the real printer, since the rogue AP has been closed.

Remote processing is executed on the cloud service (Phase 4 in Figure 6), and upon completion a notification is sent to the

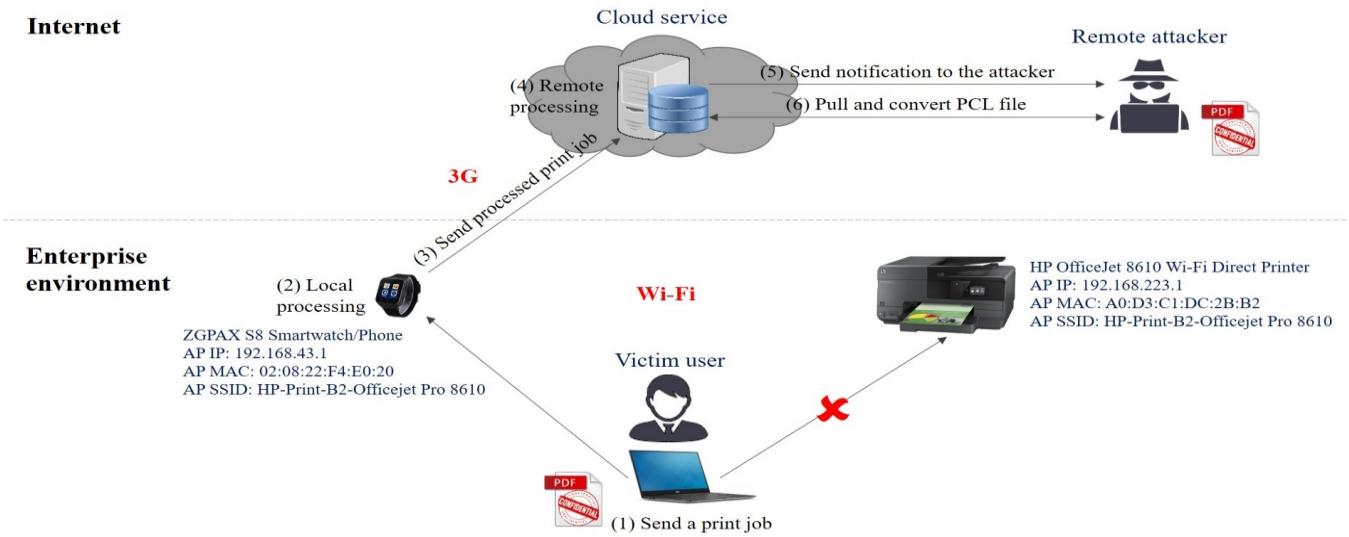


Figure 6: Data leakage attack scenario using compromised smartwatch device.

attacker, possibly by sending an email to an email account that was also defined in advance by the attacker, informing him/her about the new print job intercepted by the smartwatch (Phase 5 in Figure 6). The attacker then pulls the received processed PCL file (.txt) from the cloud service and converts it into a readable document (e.g., a PDF file), using a PCL converter (such as RedTitan EscapeE software, illustrated as Phase 6 in Figure 6).

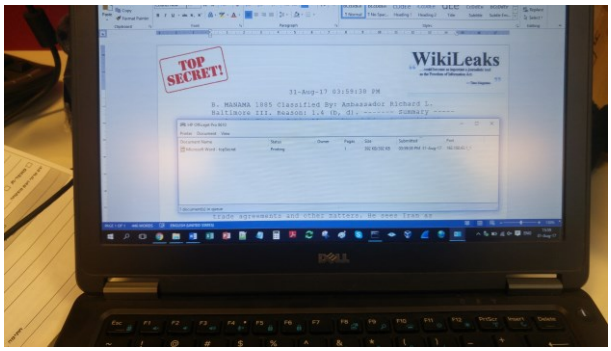


Figure 7: The status and progress of the processed print job by the smartwatch, as shown on the victim user's laptop.

3.2 Scenario Testing and Evaluation

In order to evaluate the attack scenario, the following tests were conducted: (1) Connectivity tests, evaluating the signal strength vs. distance and leap point between the real and rogue APs; (2) Performance tests of the real and rogue APs, evaluating the number of possible simultaneous connections, the maximum file size that can be processed, and the total time of the attack. All tests were conducted with both the HP OfficeJet 8610 Wi-Fi direct printer and the ZGPAX S8 smartwatch/phone device (that runs the fake printer service).

3.2.1 Connectivity Tests

In this part of our evaluation we tested the connectivity to both APs, the legitimate AP of the HP OfficeJet 8610 Wi-Fi direct printer with an IP address of 192.168.223.1 and the rogue AP deployed using the compromised ZGPAX S8 smartwatch/phone device with an IP address of 192.168.43.1. The purpose of these tests was to examine the locations from which the attack can be executed by the compromised smartwatch seamlessly without deactivating the real printer in advance. In this case, the tests are based merely on the signal strength and range from both access points, and by performing a ping connectivity test between the laptop (that connected to the AP) and the tested AP. To that end, both access points were tested, the Wi-Fi direct printer was placed in the lab in a fixed position, as shown in Figure 5, whereas the smartwatch was placed in various locations illustrating the typical operation mode of a user who wears the smartwatch on his/her wrist and wanders around the office.

The signal strength and range tests were conducted using a Dell laptop (Dell Latitude E7250 i7-5660U 2.60Ghz, 16GB RAM) with dual boot operating systems, including Windows 10 64-bit, running the Homedale Wi-Fi/WLAN monitor tool [24], and Kali-Linux Rolling 64-bit V3.22.2 using the 'airodump-ng' tool (as part of the 'aircrack-ng' package [21]). The tests were conducted on both OSs in order to cover most possible options in enterprise environments. Regular users in companies largely work with Windows-based OSs, whereas IT personnel often use Linux-based OSs for their daily tasks.

The main tests that we conducted were as follows. In the first test only the AP of the HP OfficeJet 8610 Wi-Fi direct printer was opened, and the tester moved the laptop gradually, increasing its distance from the printer (in the range of 0 to 50 meters). Next, we closed the AP of the printer and performed an identical test with the AP of the ZGPAX S8 smartwatch/phone device placed in the same place as the printer (shown in Figure 5). In both cases, as expected, the signal strength gradually diminished as the tester with the laptop moved away from the AP the laptop is connected to, until the laptop was disconnected from the AP due to low signal strength. These tests measured the effective connectivity distance without any interference. In both cases, the laptop was disconnected from the AP at a distance between 25 and 30 meters from the AP, when the signal strength was -79dBm in the case of

the real Wi-Fi printer and -86dBm in the case of the smartwatch. See Table 1 for the results obtained in these tests. Both APs behave quite similarly in terms of Wi-Fi signal intensity. When testing with Windows 10 OS we noticed that after disconnection, as soon as the tester returned to a distance of less than 30 meters, the laptop was automatically reconnected to the AP. This is because the 'Connect Automatically' feature of the Wi-Fi connection is enabled by default in Windows 10 for ease of use. In the Kali-Linux OS, on the other hand, even when the tester gets closer to the AP (~1 meter), the laptop did not automatically reconnect to the AP unless it was preconfigured manually.

Table 1: Laptop distance from AP and signal strength

Laptop distance from AP (m)	Printer AP (dBm)	Smartwatch (SW) AP (dBm)	Connectivity (printer AP/ SW AP)
0	-21	-32	+/+
5	-41	-42	+/+
10	-51	-54	+/+
15	-63	-65	+/+
20	-68	-77	+/+
25	-73	-80	+/+
30	-79	-86	-/-
35	-85	-88	-/-
40	-87	-89	-/-
45	-88	NA	-/-
50	NA	NA	-/-

The next test was to open both APs and compare the connectivity by placing them in the same position in our lab (shown in Figure 5). We then repeated the aforementioned connectivity test by moving the laptop around the lab. We found that when manually selecting the SSID name (the 'HP-Print-B2-Officejet Pro 8610') on the laptop at the start of the examination, in some cases the laptop was connected to the printer's AP, and in other cases it was connected to the smartwatch rogue AP indiscriminately.

Moreover, once the laptop was connected to one of the APs, it stayed connected to that AP throughout the entire test, until it was disconnected from the AP because it was out of range and there was lower signal strength (~25 to 30 meters away from the AP and ~-81 to -83dBm). In the case of Windows 10, in several tests we found that whenever the tester holding the laptop moved closer to the APs the laptop was automatically reconnected to the other AP, rather than the one it was connected to at the beginning of the test. This is a very important observation in that it simplifies the suggested attack model, since the victim user can be reconnected to the rogue AP of the smartwatch without requiring the attacker to explicitly deactivate the printer in advance.

The next test was to connect the laptop to the HP OfficeJet 8610 Wi-Fi direct printer and move the laptop around the lab when the ZGPAX S8 smartwatch/phone device is open and placed on the tester's wrist, imitating natural mobility. In this test the laptop and smartwatch device were in close proximity throughout the entire test as the tester holding the laptop wore the smartwatch on his/her hand. As in previous tests, the tester moved away from the printer until reaching a distance from the printer at which the laptop was disconnected from the printer's AP and automatically reconnected to the rogue AP of the smartwatch (the distance from the printer was about 25 meters). At that point, the signal strength of the printer's AP was -77dBm, and the signal strength of the smartwatch's AP was -35dBm, as shown in Figure 8. This test illustrated a normal mode of operation of disconnecting and reconnecting to Wi-Fi access points in the network with the same SSID name, namely a roaming operation, typically expected from the Wi-Fi network infrastructure in order to provide seamless and continuous connectivity. However, in this case the roaming is to the rogue AP of the smartwatch device. Again, this is an important observation about our suggested attack model, showing that explicit printer deactivation is not a prerequisite for a successful attack.

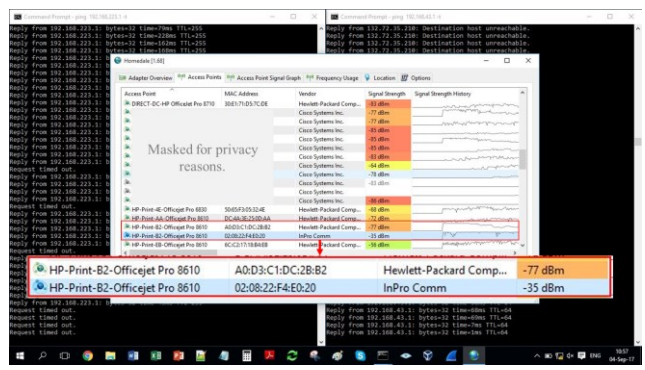


Figure 8: Roaming between the printer's AP and the smartwatch's AP.

The last two tests were quite similar, where both the APs of the HP OfficeJet 8610 Wi-Fi direct printer and the ZGPAX S8 smartwatch/phone device were open, and the printer and the smartwatch were stationary, at a distance of 40 meters from each other. The tests include connecting the laptop (with Windows 10) to one of the APs and moving toward the other AP. In other words, the tester holding the laptop moved away from the AP he/she was connected to and moved closer to the other AP. In the case of first connecting to the HP OfficeJet 8610 Wi-Fi direct printer's AP and walking in the direction of the ZGPAX S8 smartwatch device's AP, the leap between the APs occurred at a distance of about 25 meters, with signal strengths of -79dBm for the printer's AP and -57dBm for the smartwatch's AP, as illustrated in Figure 9. In the other case, where the laptop was first connected to the ZGPAX S8 smartwatch/phone device's AP, and the tester walked in the direction of the HP OfficeJet 8610 Wi-Fi direct printer's AP, the results were nearly the same: the leap between the APs occurred at a distance of about 25 meters, with signal strengths of -81dBm for the smartwatch's AP and -68dBm for the printer's AP. This test marked the point at which the roaming between the APs occurs automatically without any intervention from the attacker.

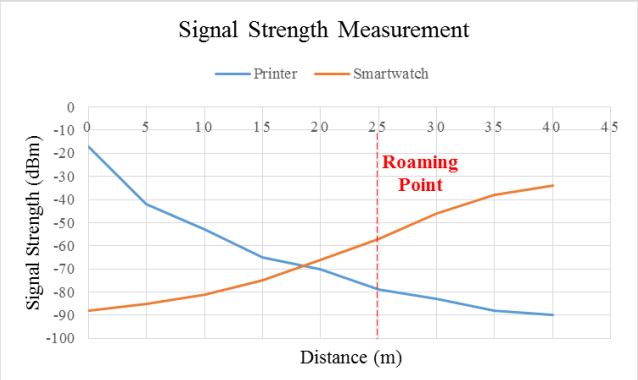


Figure 9: Roaming point between access points of the printer and smartwatch.

3.2.2 Performance Tests

The next tests evaluated several performance parameters, including the maximum number of simultaneous device/user connections that can be supported by both APs; the capacity of the printer queue and the maximum file size that can be handled by the fake printer; and the total time of the attack executed by the smartwatch device with respect to several file sizes. These tests attempted to determine the performance limitations of the fake printer service running in the network and thus of the suggested attack model.

Regarding the maximum number of simultaneous connections, in the case of the HP OfficeJet 8610 Wi-Fi direct printer it was found that at most five devices can be connected simultaneously to the printer (also defined in the functional specifications of the printer[1]). In the case of the ZGPAX S8 smartwatch/phone device eight simultaneous user/device connections can be supported (as shown in Figure 10). During the test we were able to connect different clients to the smartwatch's AP (several smartphones and laptops), and our findings imply that potentially an attacker can simultaneously steal confidential and private information from up to eight different connected users in the enterprise network using the smartwatch as the attack means.

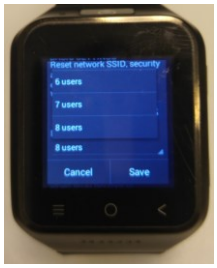


Figure 10: Maximum simultaneous device/user connections in the smartwatch's AP.

Next, we evaluated how the Wi-Fi printer service queue of the fake printer handled multiple print jobs and determined the maximum file size that could be supported by the smartwatch. Accordingly, several documents (.pdf files) with different file sizes (between ~10KB and ~50MB) were sent to the fake Wi-Fi printer service running on the smartwatch. We found that in most cases the fake Wi-Fi printer service actually handled these print jobs well. Figure 11 contains a list of all of the received and processed print jobs during our attack (the .pdf files that were converted to the PCL data format as .txt files).

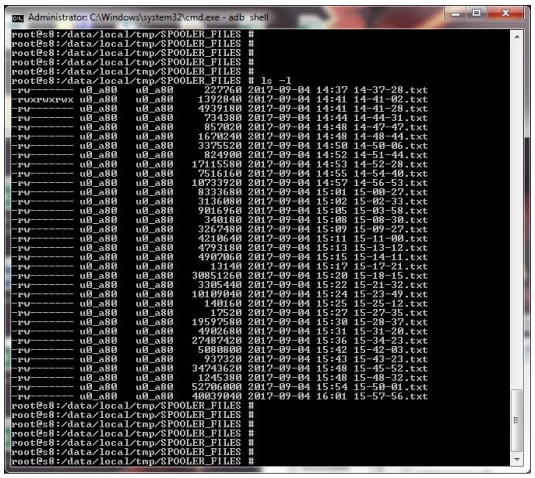


Figure 11: The list of print jobs processed by the smartwatch shown via an Android Debug Bridge (ADB) connection.

Next, the total time of the attack executed by the smartwatch device when the laptop was placed ~1 meter from the smartwatch was measured using files of various sizes (between ~500KB and ~35MB, see Table 2). The total attack time was measured with respect to: (1) the time it takes to open the malicious application and run the rogue access point on the smartwatch device (about 28 seconds), (2) the time it takes the laptop to connect to the fake printer (connect to the rogue AP, about 37 seconds), (3) the processing time by the smartwatch per print job as a function of the print job file size (as shown in Table 2), (4) the time it takes to pull the file from the smartwatch as a function of the print job file size (as shown in Table 2), and (5) the time it takes to close the rogue AP by the malicious application (~10 seconds). We did not consider the time it took to convert the file to a readable format, since this is software dependent. We found that for a file of 640KB the total time of the attack is 90.82 seconds, whereas for a file of 35MB the total attack time is 168.42 seconds. This time mainly depends on the processing and download times for each file as depicted in Table 2. These results are significant, because in order to avoid suspicions by the victim user or IT security personnel, the compromised smartwatch device should close the fake printer service within a reasonable time interval.

We concur from the above tests that the fake Wi-Fi printer service running on the compromised smartwatch device is, in most cases, a viable means of leaking print jobs from the organization. Typical users, even professionals like network administrators, are unlikely to notice that something is wrong or suspicious when print jobs are leaked to the fake printer. The suggested attack model and its practical proof of concept illustrate the potential and overlooked risks posed by the use of smartwatch devices in enterprise environments. A demonstration of the attack scenario can be found in the following link: https://youtu.be/FiEwqfb63IY.

[1] http://www8.hp.com/uk/en/campaigns/wireless-printing-center/wireless-direct.html

Table 2: Total time of the attack for several file sizes

File size on smartwatch (MB)	Processing time (Sec)	Download time (Sec)	Total time of attack (Sec)
0.64	15	0.82	90.82
2	18	0.85	93.55
3.75	22	1.3	98.3
5.7	29	1.76	105.76
8.17	32	2.28	109.28
10.7	46	3.1	124.1
14.8	60	4.08	139.08
20.8	66	4.47	145.47
31.3	77	8.4	160.4
35	86	7.42	168.42

4. ATTACK MITIGATION AND COUNTERMEASURES

In this section we propose an attack mitigation process and countermeasures for the specific attack scenario presented in this paper. The suggested detection and prevention techniques are aimed at dealing with the attacks executed in the network by the compromised smartwatch device (i.e., reconnaissance and rogue access point attacks). The suggested techniques include: (1) conducting a preliminary security acceptance testing procedure before allowing the use of the smartwatch device (and IoT devices in general) in the organizational network in order to assess the security risks of the device to the organization; (2) performing real-time network traffic monitoring and peripheral scanning of the wireless Wi-Fi access points deployed in the network, mainly to detect anomalous behavior obtained by the smartwatch device in the network; and (3) proactively performing prevention operations in order to limit the capability of the remote attacker to leak/steal sensitive information from the organization, as described in the following sections.

4.1 Security Acceptance Testing

In general, in order to deal with the proliferation of personal IoT devices connecting to the enterprise network, where more and more employees bring their own personal mobile devices (e.g., tablets, smartphones, and smartwatch devices) to their workplace as part of the Bring Your Own Device (BYOD) trend, one may consider performing a security acceptance testing procedure before allowing the use of such devices within the organizational network [25, 26]. This can be conducted using a security testing process targeted specifically at mobile and wearable IoT devices [19]. The sysadmin and/or security officer of an organization may perform security acceptance testing to determine if the IoT device under test complies with a set of predefined security requirements and determine whether the device is compromised by malicious applications before allowing the device to be connected to the network.

With regard to the attack scenario presented in this paper, as part of the security acceptance testing procedure, a preliminary security analysis is conducted for the ZGPAX smartwatch device, including a device feature and capability test, a known vulnerabilities assessment test, and a user applications analysis test (which is aimed at detecting malicious applications running on the tested smartwatch by analyzing each application installed on the device). Forensic analysis is then performed in order to determine the security risk associated with using the ZGPAX smartwatch device in an enterprise environment.

4.1.1 Device Feature and Capability Test

In this security test we examined the set of features and capabilities of the ZGPAX smartwatch device under test in order to determine the possible security risks of the smartwatch to the organization. To that end, we tested the device configuration, and we found that it includes GPS, camera, and voice recording features along with a variety of network connectivity capabilities, including several wireless communication channels, such as Wi-Fi, Bluetooth, and cellular network connectivity (by using an appropriate SIM card with the device), as well as wired connectivity with a micro USB slot where ADB (Android Debug Bridge) connection can be enabled. Table 3 summarizes the device features and capabilities along with possible threats and attack vectors for using the ZGPAX smartwatch device as a means of attack.

From this security test it can be concluded that the ZGPAX smartwatch device is highly accessible for attackers via different types of network connectivity (wireless and wired) and has advanced features and capabilities (GPS, camera, and voice recording). Thus, it is prone to various remote and local attacks and can be used as a means of attack in enterprise networks (as was shown by the attack scenario presented in this paper); therefore, it should be considered a high risk to the organization.

Table 3: ZGPAX smartwatch device features and capabilities and possible threats and attack vectors

Device Feature/ Capability	Possible Threats and Attack Vectors
GPS	Tracking: location and positioning detection.
Camera	Privacy violation: video recording and face recognition.
Voice Recording	Privacy violation: recording private conversations.
Wi-Fi Connectivity	Mid-range connectivity: internal IP network attacks (e.g., network mapping, rogue access point, data leakage).
Bluetooth Connectivity	Short-range connectivity: local network attacks.
Cellular Network Connectivity	Long-range connectivity: remote network attacks.
Micro USB Port	Physical device connectivity: access for data at rest.

4.1.2 Known Vulnerabilities Assessment Test

In this security test we assessed the security level of the ZGPAX smartwatch device by analyzing the set of known vulnerabilities associated with the device. To that end, we employed both Nessus [27] and Retina [28] vulnerability security scanners. From the results obtained, shown in Figure 12, we found that five known

vulnerabilities exist in the ZGPAX smartwatch device, with only two CVE records (Common Vulnerabilities and Exposures), including CVE-2003-0001 with a low (3.3) severity/risk score (as defined by NVD CVSS version 2 calculator [29]), and CVE-1999-0524 which is only an info level vulnerability (there is no risk). Based on these results it could be concluded that the ZGPAX smartwatch device has low risk. However, due to the limited amount of information about the known vulnerabilities associated with the ZGPAX smartwatch device, this factor (as part of the security acceptance test) should have little influence on the decision whether to allow the operation of the tested smartwatch device in the network made by the sysadmin/security officer of the organization.

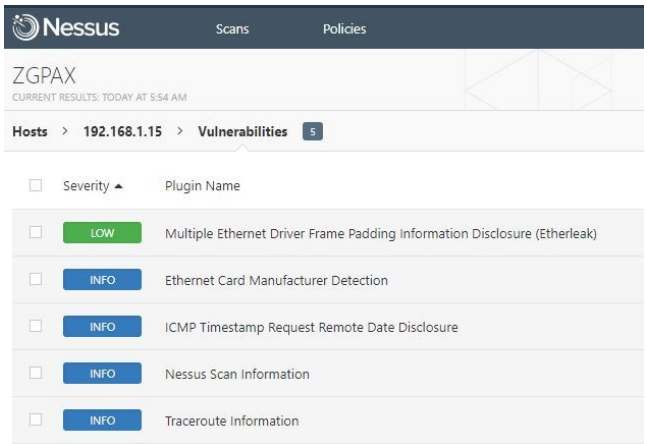


Figure 12: Vulnerability scanning for the ZGPAX smartwatch using the Nessus vulnerability scanner tool.

4.1.3 User Applications Analysis Test

In this security test we examined the user applications installed on the ZGPAX smartwatch device in order to detect malicious applications running on the device. To that end, we ran and analyzed each user application installed on the smartwatch device separately by utilizing the Nmap security scanner tool [20]. In order to reveal unusual behavior (i.e., illegal open ports for that application), for each user application we examined the list of all ports opened by the application when it was activated on the smartwatch device.

From this security test we found that the smartwatch device is installed with several built-in user applications (provided by the manufacturer of the smartwatch device), such as People (contact list), Browser, Calculator, Calendar, Facebook, Gmail, Maps, and more, and that all of these user applications operated normally (no abnormal behavior is obtained from the open port perspective). The only user application that attracted our attention is the MyClock application - once we activated this application we found that it opens the TCP port 9100 which is the Jetdirect service of the HP printer, as shown in Figure 13. An application that opens a TCP/UDP port or service which is not in its contexts (e.g., the MyClock application should not open a printer service and ports) is demonstrating unusual behavior, and thus it should signify that a malicious application is running on the tested smartwatch device. Accordingly, from the analysis conducted, the sysadmin/security officer should define this smartwatch device as a high risk to the organization.

This illustrates how malicious applications running by the ZGPAX smartwatch device can be detected as part of the suggested security acceptance testing procedure. Note that the MyClock application is the one we implemented and installed on the smartwatch device in order to prove the feasibility of the attack scenario presented in this paper. This process should be followed as part of the preliminary security acceptance testing in order to detect malicious applications running by any IoT device in advance.

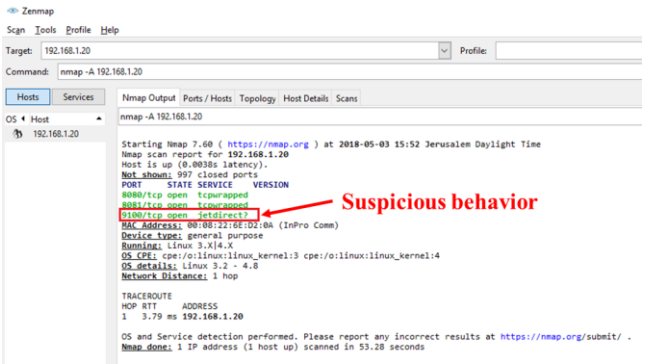


Figure 13: Detecting the TCP port 9100 Jetdirect service running by the malicious application in the ZGPAX smartwatch device using the Nmap security scanner tool.

To conclude, based on the security acceptance testing procedure performed and described above, including a device feature and capability test, a known vulnerabilities assessment test, and a user applications analysis test, the sysadmin/security officer should consider the tested ZGPAX smartwatch device as a high risk to the organization, and therefore prohibit the use of such a device on the network (particularly if the malicious application is detected during the acceptance testing procedure). In cases in which the smartwatch is allowed to be connected to the network by the sysadmin/security officer or a device has not undergone security acceptance testing (a situation which is common in real-life scenarios), additional mitigation mechanisms and countermeasures should be employed within the organization in order to detect and prevent possible attacks in the network by compromised IoT devices.

4.2 Network Monitoring and Scanning

Real-time network traffic monitoring along with peripheral scanning of the wireless Wi-Fi access points deployed in the network should be performed regularly by the IT department (sysadmin/security officer), mainly for detecting anomalous behavior in the network. Using this approach, both the network scanning/mapping attack (conducted by the compromised smartwatch device during the reconnaissance phase of the suggested attack model), and the rogue access point attack (executed by the smartwatch during the leakage phase of the attack model), can be detected and defined as abnormal behavior in the network, as presented in the upcoming sections.

4.2.1 Network Traffic Monitoring

Network traffic monitoring and analysis can be conducted using various network scanning tools. By utilizing advanced tools and techniques this operation can be performed automatically and in real-time. For this specific proof of concept we recorded, analyzed, and detected anomalies obtained in the network

manually. For simplicity, we used Wireshark [22] in order to collect the network traffic in our internal network. First, all of the network traffic was recorded and stored as *.pcap files. Next, we investigated and analyzed the network traces from the recorded files (*.pcap) in order to detect abnormal behavior in the network. Finally, forensic analysis was performed in order to determine whether the detected anomalies are actually attacks performed in the network and identify the source.

To that end, the compromised smartwatch device was connected to our internal network in the lab in order to monitor its network activity during this test. From the analysis conducted, it was found that the smartwatch device sends bursts of ARP (Address Resolution Protocol) requests at some point of time in the examination, characterizing a network scanning and mapping operation, as illustrated in Figure 14. Therefore, using this security analysis, we can actually declare that we managed to detect the network reconnaissance attack which was executed by the compromised smartwatch device as part of the suggested attack model.

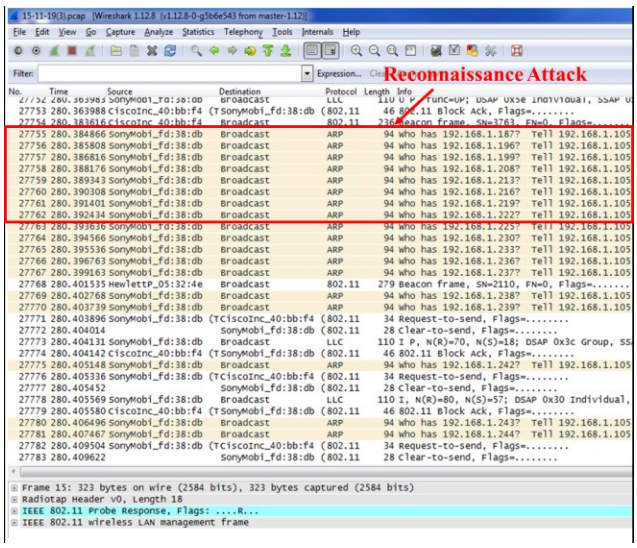


Figure 14: Detecting bursts of ARP requests sent by the compromised smartwatch device during the reconnaissance phase of the suggested attack model.

4.2.2 Peripheral Scanning of Wi-Fi Access Points

Wireless Wi-Fi access points, or for short, access points (APs), are deployed in the organization in order to enable seamless continuous wireless network connectivity mainly to mobile computing devices, such as laptops, smartphones, tablets, and smartwatch devices, which connect to the network (stationary PC stations with wireless network cards can be connected to the APs as well). This allows the device to always be connected to the network using the roaming process of Wi-Fi connectivity. In addition, different Wi-Fi direct services are employed in the network, such as Wi-Fi direct printers, enabling ad hoc connection between devices in the network without the need for a wireless router. From that perspective, several APs, each configured with a unique SSID (Service Set Identifier) name that identifies the AP in the network, are operated in the organization, some for wireless network connectivity and others for ad hoc Wi-Fi direct services (mainly printing services).

The most common attack in such infrastructure/deployments is the installation of rogue wireless access points in the network [30]. The rogue AP used as a malicious AP service that is configured with the same SSID name as an existing AP in the network, either by an internal employee or by an attacker (internal/external), in order to execute different types of attacks on the organization (e.g., data leakage attacks). Thus, in order to detect rogue wireless APs in the network, the sysadmin/security officer of the organization should first map the locations of all wireless Wi-Fi access points deployed in the network in advance, as in most cases APs are installed in stationary positions in the organization. Once this has been done, peripheral scanning of the APs deployed in the network should be performed regularly in order to detect malicious behavior in the network.

Considering the attack scenario in this paper, peripheral scanning of wireless APs was executed using several security tools, including the Homedale Wi-Fi/WLAN monitor tool [24] and 'airodump-ng' tool (as part of the 'aircrack-ng' package [21]), in order to monitor the radio spectrum in our lab (shown in Figure 15). Using this approach, the rogue access point opened by the ZGPAX compromised smartwatch device during the leakage phase of the suggested attack model was detected. Note that two APs with the same SSID name transmitting with the same channel number and the same signal strength cannot operate in the same physical location in the organization; this either represents inadequate network planning and management, or signifies a rogue access point attack, as in our case. In addition, since the sysadmin/security officer maps all of the APs deployed in the network in advance, an AP that operates in unusual locations in the organization should also be considered a rogue AP (in our case, this refers to the scenario where the compromised ZGPAX smartwatch device is running its rogue AP in locations other than where the Wi-Fi HP direct printer is located). Accordingly, once this anomaly is detected during the peripheral scanning of wireless APs, we can declare that we were able to detect the rogue access point attack executed by the compromised smartwatch device in the network. This is obtained by looking at the BSSID name (the MAC address) in the results shown in Figure 15 (where the MAC address of the HP Wi-Fi direct printer AP is A0:D3:C1:DC:2B:B2, and the MAC address of the ZGPAX smartwatch device AP is 02:08:22:F4:E0:20); this is also shown in Figure 6.

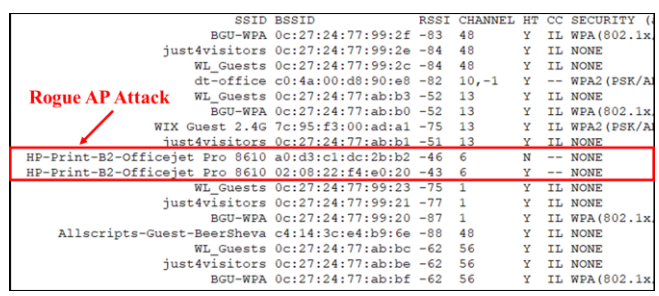


Figure 15: Detecting the rogue access point attack executed by the compromised ZGPAX smartwatch device using peripheral scanning of wireless APs deployed in the network.

To conclude, by performing network traffic monitoring and peripheral scanning of the wireless Wi-Fi access points deployed in the network, suspicious behaviors were detected, which can be

considered as anomalous behavior in the network related to the attacks executed by the ZGPAX smartwatch device as part of the attack model presented in this paper. In order to address these issues, additional mitigation mechanisms and countermeasures should be employed in the organization in order to proactively prevent possible attacks in the network.

4.3 Proactive Attack Prevention

A proactive attack prevention procedure should be performed by the sysadmin/security officer of the organization as soon as abnormal behavior has been detected in the network. Once the specific devices causing the anomalies are identified (during the network and monitoring scanning attack mitigation process), prevention techniques must be employed in order to avert the execution and/or completion of the attack before real harm is caused to the organization. In this paper, we proposed two proactive prevention methods to deal with the possible attacks presented in the attack model (i.e., reconnaissance and rogue access point attacks), including device quarantine and isolation, and access point deactivation, which are discussed in the upcoming sections.

4.3.1 Device Quarantine and Isolation

Device quarantine and isolation is the process of isolating suspicious devices, both standard computing stations, such as PCs and laptops, and also IoT devices, from the network in order to minimize the harm that these devices can cause to the organization. When anomalous behavior has been detected by the monitoring process, the suspicious/compromised device causing this behavior should be immediately disconnected from the main network in order to prevent the execution of possible attacks in the network. This operation can be executed manually by the sysadmin/security officer once an alert is initiated, or automatically by employing an adaptive network access quarantine control (NAQC) solution targeted for such situations.

Considering the attack scenario presented in this paper, once a reconnaissance attack is executed by the compromised smartwatch device and detected by the network monitoring process, the device which was connected to the internal Wi-Fi network of the organization (e.g., subnet 192.168.1.0/24), will be manually/automatically disconnected and isolated from the network (e.g., it will be connected to an isolated subnet with IP address 10.0.0.1), as illustrated in Figure 16. The smartwatch should remain disconnected from the main network until further investigation is performed by the sysadmin/security officer as part of an advanced security forensic analysis process which includes in depth inspection and examination of the device's activities in isolated environments, such as a sandbox or security testbed for IoT devices [19].

4.3.2 Access Point Deactivation

In cases in which the device has the capability of Wi-Fi direct service, no connection to the internal networking infrastructure (e.g., router) is needed in order to establish connectivity between devices in the network. Whereas the device is not connected internally to the network, it is physically deployed in the network environment and opens an independent access point (e.g., Wi-Fi direct printer). In such situations, an additional prevention solution (besides device quarantine and isolation) is required.

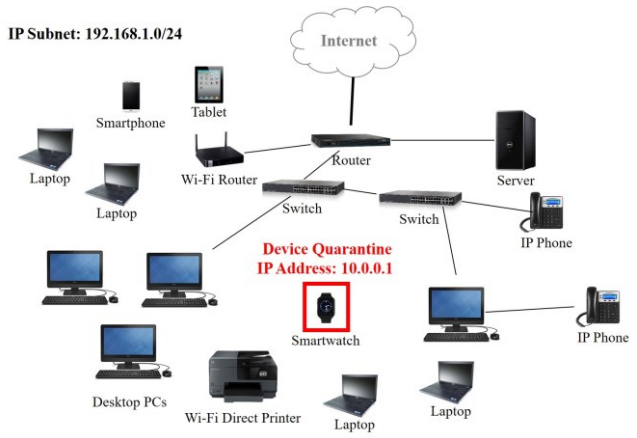


Figure 16: Illustration of the ZGPAX smartwatch device isolation from the network.

During the peripheral scanning of wireless AP procedure, once the abnormal behavior of the compromised ZGPAX smartwatch device is detected, an access point deactivation process should be executed proactively and immediately, either automatically by the IDS/IPS (Intrusion Detection/Prevention System) solution deployed on the network, or manually by the sysadmin/security officer of the organization, in order to mitigate this rogue access point attack. The detection of abnormal behavior is an indication that that the device has opened a rogue access point (as part of the leakage phase of the attack model) with the same SSID name as the HP Wi-Fi direct printer (namely 'HP-Print-B2-Officejet Pro 8610') and transmitted on the same channel number (in our case, channel 6, as shown in Figure 15) and with nearly the same signal strength as the HP Wi-Fi direct printer.

In this paper we illustrate the deactivation of access points, and specifically the rogue access point opened by the compromised ZGPAX smartwatch device, using the 'aireplay-ng' tool (as part of the 'aircrack-ng' package [21]). Namely, before the data leakage attack is executed by the compromised smartwatch, deauthentication packets are sent to the smartwatch device, targeted to the BSSID name (MAC address) of the rogue access point running by the compromised smartwatch as illustrated in Figure 17, in order to deauthenticate all connected clients and disable the connection of that access point in the network. Using this approach, upcoming print jobs will not be processed by the compromised smartwatch device, thus no information will be leaked from the organization (using the smartwatch as a means of attack).

5. RELATED WORK

Smartwatch adoption has increased in recent years due to the technology's evolvement and a steady price decline. Current smartwatch models are equipped with various sensors that enable different real-time measurements of their users with a high degree of accuracy. Such capabilities can be positively utilized for a variety of purposes. As a case in point, Lewis et al. [31] proposed a motion-based real-time authentication scheme that applies behavioral biometrics to verify a smartwatch user by collecting and analyzing several metrics obtained from the embedded motion sensors of a smartwatch.

On the other hand, several works have demonstrated ways in which smartwatch devices can be used for attacks. For instance, Migicovsky et al. [32] highlighted a security risk using the Pebble smartwatch by developing a malicious application that enabled students to collaborate and cheat on multiple-choice exams. Although no private information was leaked, the authors demonstrated the usage of a simple smartwatch device in a novel attack scenario. Another attack scenario utilizing a smartwatch device was presented by Denney et al. [33]. The authors demonstrated a covert channel attack on an Android smartwatch device. The attack used a storage covert channel that can be implemented on wearable devices to leak data by utilizing status bar notifications.

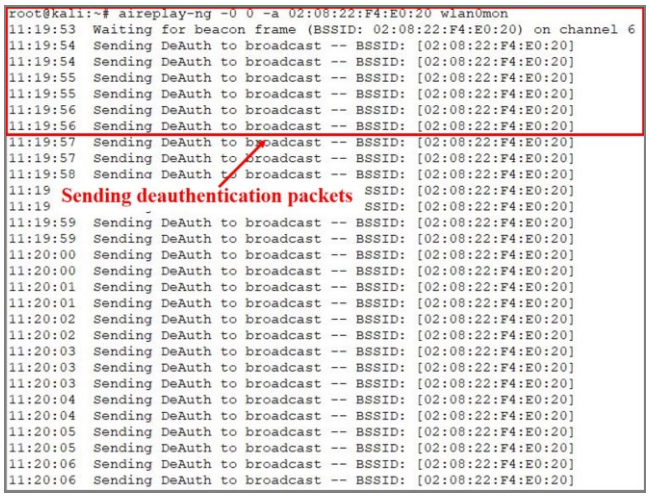


Figure 17: Illustration of the deactivation of rogue access point running by the compromised ZGPAX smartwatch device using the aireplay-ng tool.

Several side-channel attacks using smartwatches utilize the embedded motion sensors, specifically the accelerometer and gyroscope, as means for keystroke inference attacks. For example, Wang et al. [34] developed Motion Leaks (MoLe), a completely functional system on Samsung Gear Live smartwatches, in order to leak information about what the user is typing on a regular (laptop or desktop) keyboard while wearing a smartwatch device. Combined with patterns in the English language, the accelerometer and gyroscope data was processed through a sequence of steps, including key press detection, hand motion tracking, character point cloud computation, and Bayesian modeling and inference. Keystroke inference attacks on a smartphone's numeric touchscreen keypad while wearing a smartwatch were presented by Maiti et al. [35]. The proposed attack approach employs supervised learning techniques using appropriate classification models, such as simple linear regression (SLR), random forest (RF), and k-nearest neighbors (k-NN) algorithms, in order to accurately map the uniqueness in the captured wrist movements to each individual keystroke. The authors used a Samsung Gear Live smartwatch equipped with a gyroscope, accelerometer, and compass sensors in their evaluations. Another side-channel attack that aims to infer user inputs on keyboards by exploiting sensors in the smartwatch was proposed by Liu et al. [36]. A malicious application was developed and installed on the LG G watch, which recorded the acoustic signals and accelerations simultaneously. The raw acceleration data was processed and converted into features which were fed into a modified k-NN algorithm for training a model, which was then used in the attacking phase to reproduce a set of PIN (personal identification number) candidates ranked by their possibilities. The authors focused on two major categories of keyboards: the numeric keypad which is generally used to input digits (e.g., using a POS terminal to enter banking PINs), and the QWERTY keyboard on which a user can type English text. Sarkisyan et al. [37] demonstrated a method for inferring smartphone PINs while wearing a compromised smartwatch (a malware accessing only the smartwatch's motion sensors, including the accelerometer and gyroscope, was used as a side-channel attack). A random forest classifier was employed in order to infer the PINs entered by the user based on unlabeled sensor data collected from the smartwatch during a specific time interval.

Several authors have discussed inherent security and privacy risks of mobile and wearable smart IoT devices in general. For instance, Torre et al. [38] presented a case study about privacy violations by exploiting data obtained from fitness trackers available to a third party as authorized by the user (i.e., sensor data and profiling data provided by the user while registering for the service). The authors implemented a reasoner using a Bayesian network which was initialized with prior probabilities and conditional probabilities, for parent and child nodes, respectively. Thus, it was able to simulate the posterior probabilities which represent the risk that a third party might infer a private data item given the set of shared data. Nahapetian [39] highlighted several possible side-channel attacks on mobile and wearable devices, specifically using a malware that passively exploits sensors embedded in such devices in order to monitor and collect sensitive personal information. The side-channel attacks in this case aim to capture sensitive private information, such as PINs, passwords, patterns for unlocking smart phones, entered text, surrounding private information, such as conversations or neighboring computing activity, and more. Suarez-Tangil et al. [40] discussed the inherent security risks of using third party applications that can be downloaded and installed on different IoT mobile devices. They pointed out that this approach can lead to covert malware installation causing severe security breaches. Their work, however, is mainly focused on smartphones and tablets and does not consider smartwatch devices. Jiajia Liu and Wen Sun [41] discussed possible attack methodologies on smart wearable devices, including smartwatches, from the perspectives of privacy, authenticity, and data integrity. For example, from the privacy perspective, the motion sensors of a smartwatch could leak personal information; attacks against authenticity in wearables include manipulation of the sensing environment and the wearable sensor itself; and from data integrity aspect an adversary could eavesdrop on the data communication between wearable devices and their paired device (mainly smartphones) due to low encryption mechanisms (e.g., Bluetooth Low Energy, BLE, case).

With respect to integrating mobile IoT devices in enterprise environments, most of the work on security risks (Singh [8], Koh et al. [9]) has focused on the emerging BYOD trend pertaining to smartphones and tablets but does not address the specific characteristics of smartwatch devices. The work points out the benefits of bringing a personal smart device to work (increased productivity and flexibility) and acknowledges the risks, noting

that BYOD devices might expose the enterprise to numerous security breaches. Furthermore, Nurse et al. [18] presented the implications of using IoT devices, and specifically personal devices, to facilitate insider leakage within enterprises. The authors review the range of new and adapted attack vectors opened by IoT mobile devices, including smartwatch devices in enterprise networks, but ignore specific attacks potentially targeted on Wi-Fi printers.

A man-in-the-middle (MITM) attack on Wi-Fi networks is discussed in detail by Vanhoef and Piessens [42]. The authors presented various attack scenarios using off-the-shelf Wi-Fi dongles, none of which considered attacking Wi-Fi direct printers. Monica and Ribeiro [43] presented the security risks associated with the evil twin attack in Wi-Fi networks, where a rogue AP mimicking a legitimate AP in the network is established to eavesdrop on wireless communications. Their work focuses on Wi-Fi hotspots with Internet access, again without specifically considering Wi-Fi direct printers.

From the perspective of mitigation and countermeasures for the BYOD issue, several studies were conducted. Morrow [25] introduced security concerns associated with mobile device vulnerabilities in enterprise environments, and proposed safeguards and proactive steps to ensure that every mobile device in the organization has the appropriate security controls installed. Miller et al. [26] presented security and privacy considerations for the BYOD trend, and the authors emphasized the need to enforce policies in such situations as the key factor for dealing with BYOD concerns. Several security countermeasures were suggested by Tokuyoshi [44] in this context, including protecting the network traffic and the content of the traffic from vulnerabilities and exploits, enforcing application policy and device policy, and protecting the data on the device. Eslahi et al. [45] presented an overview of the current state of BYOD security solutions, referring to three main security models including Mobile Device Management (MDM), Mobile Application Management (MAM), and Mobile Information Management (MIM) models. An analysis of the key risks related to BYOD security threats along with how existing control mechanisms can be used to address these threats was discussed by Rivera et al. [46]. The main conclusion obtained by the authors is that a secure adoption of BYOD requires a combination of technical and nontechnical security controls to be established in the organization. Costantino et al. [47] proposed a role-based access control system which is based on user identity and the user's current context to enforce on the fly instantiated policies inside organizations. Further risk assessment for BYOD was conducted by Tanimoto et al. [48] using risk management methods, including Risk Breakdown Structure (RBS) and risk matrix. They found that the main risk of the introduction of BYODs to an enterprise is a leak of information, and that Mobile Device Management (MDM) is shown to be the most effective countermeasure in such situations. Ratchford et al. [49] discussed the legal and privacy-related issues that organizations may encounter when adopting BYODs and proposed recommendations in order to mitigate these types of risks.

With respect to the reconnaissance and rogue access point attacks presented in this paper, different solutions have been proposed. Uma and Padmavathi [50] presented a survey on various cyber-attacks and their classification, including reconnaissance attacks and their implications, in order to create awareness so that appropriate defense measures can be initiated against such attacks. Chen et al. [51] proposed a defense system based on analyzing logs from the network and extracting the reconnaissance attack sequences related to targeted attacks using the adaptation of a state-based hidden Markov model in order to detect the joint attacks. The results showed that the proposed solution can efficiently identify the attacks at an early stage to prevent further damage in the networks. Beyah and Venkataraman [30] examined different mechanisms for detecting rogue access points inserted by malicious insiders and outsiders to the organization, and proposed a hybrid solution for such situations. The suggested detection method combined a rate adaptation-based detection solution to identify stable periods with a distributed coordination function (DCF) based detection solution to extract the DCF components in those periods. A rogue access point detection and localization (RAPDL) software system solution was presented by Le et al. [52]. The RAPDL system was implemented as a client-server model, where the client monitors identify potential rogue APs, measure their properties, and report relevant information to the server, whereas the server runs localization algorithms (a distance-based and a fingerprint-based algorithm) to identify and locate the rogue APs. Anmulwar et al. [53] presented different approaches of rogue access point detection, including a client-side, server-side, and hybrid approach; the latter was found as the most efficient method for the detection of rogue access points, as it minimizes the ineffectiveness of the client-side approach and adds server control to the detection mechanism.

6. SUMMARY AND FUTURE WORK

The Internet of Things exposes many new security and privacy risks to enterprises. Specifically, the abundant use of personal or wearable computing devices, such as smartphones, tablets, and smartwatch devices as part of the Bring Your Own Device trend, increases the attack surface and vulnerability of enterprises. In this research we demonstrated a novel attack scenario by exploiting a smartwatch device worn by an enterprise employee. Using standard capabilities of a ZGPAX S8 smartwatch/phone device infiltrated by a malicious application, we deployed a rogue AP with an identical SSID name as a legitimate Wi-Fi direct printer. The aim of this scenario was to illustrate how an advanced attacker located outside of the organization can use a compromised smartwatch device to unobtrusively leak sensitive and private information by stealing print jobs sent to a legitimate printer that exists in the network.

The underlying motivation for this attack scenario is that Wi-Fi direct printers are increasingly being deployed in enterprise environments [54]. Such printers provide ad hoc printing service to Wi-Fi based mobile devices (mainly laptops, smartphones, and tablets), without requiring a prior full-fledged connection to the internal networking infrastructure (such as modems/routers, APs, etc.) of the enterprise. In most cases, Wi-Fi direct printers are deployed with the default factory settings and without encryption (no password protection) for ease of use. This may increase the likelihood of security breaches and data leakage attacks, specifically in enterprise environments. In addition, we choose to focus on the possible security risk posed by using smartwatch devices in enterprise environments, since these smart connected devices become standalone devices that are supplied with Wi-Fi and/or cellular connectivity [55]; thus they are no longer

dependent on a paired smartphone and can be used as any other mobile device in the network. Furthermore, smartwatch devices are naturally integrated in the network as are they worn by employees, and are currently less controlled and monitored than standard mobile computing devices (laptops, tablets, and smartphones), hence they may become a new means for attacks and malicious activities in enterprise networks.

The attack model considered in this paper is comprised of a remote attacker that utilizes a smartwatch devices in a multi-phase attack, including: a reconnaissance phase, in which information about the Wi-Fi printers deployed in the network is gathered by the compromised smartwatch; a traffic analysis phase, in which the protocol messages exchanged between a legitimate computer and the selected Wi-Fi direct printer are analyzed and reconstructed offline by the attacker in order to accurately imitate a real Wi-Fi printer service during the attack; and a leakage phase, in which the compromised smartwatch device running a fake Wi-Fi direct printer service in the network (using a malicious application that establishes a rogue AP with the identical SSID name of a legitimate printer) in order to intercept sensitive print jobs sent to a Wi-Fi direct printer in the organization.

We evaluated the attack scenario and the capability of the rogue AP by conducting several tests, including connectivity tests, used to define the leap point between the real and rogue APs in several cases based on the signal strength and distance measures; and performance tests, used to evaluate several performance parameters, such as the number of possible simultaneous connections, the maximum file size that can be processed, and the total time of the attack. There are some limitations/weaknesses with the suggested scenario, in which the compromised smartwatch device needs to be in proximity of the attacked victim user and the printer, as well as the need to deactivate the real printer in advance such that all print jobs will be processed by the compromised smartwatch instead of the real printer. However, the results obtained from our practical proof of concept evaluation prove the strength of the suggested attack scenario by demonstrating that the fake Wi-Fi printer service running on the compromised smartwatch device is a viable method for leaking sensitive and private information in enterprise environments.

An attack mitigation process and countermeasures for the presented attack scenario were proposed in order to prevent the compromised smartwatch device from executing the attack in the network, thus preventing the leakage of sensitive information from the organization. This is done by employing several detection and prevention techniques, including: (1) a preliminary security acceptance testing procedure, using security tests such as a device feature and capability test, a known vulnerability assessment test, and a user applications analysis test, which is conducted by the sysadmin/security officer of the organization before allowing the use of the smartwatch device in the network and in order to assess the possible security risks of the smartwatch to the organization; (2) real-time network traffic monitoring along with peripheral scanning of the wireless Wi-Fi access points deployed in the network which is employed in order to detect abnormal behavior in the network, specifically those caused by the compromised smartwatch device; and (3) proactively employed countermeasure techniques, such as device quarantine and isolation, and rogue access point deactivation, which are based on the information collected during the monitoring process. Using this approach, any IoT device (not just smartwatch devices) can be tested (before allowing the device to be connected to the network), monitored (during its activity in the network), and blocked (if identified as the source of the anomalies occurring in the network), in order to minimize the harm that can be caused by these smart devices to the organization.

In future work we intend to extend our attack model and the proof of concept implementation in order to support a full man-in-the-middle attack process performed by the smartwatch device. This will include: automatically scanning the network and selecting the target Wi-Fi direct printer, disabling/deactivating the selected printer remotely, and sending the intercepted print jobs from the smartwatch directly to the legitimate Wi-Fi direct printer upon completion of the attack. Another open issue to be investigated involves dealing with password protected printers, as well as implementing, testing, and employing additional mitigation and countermeasure techniques for such possible scenarios.

7. REFERENCES

[1] Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer networks, 54(15), 2787-2805.

[2] Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. Information Systems Frontiers, 17(2), 261-274.

[3] Weber, R. H. (2010). Internet of Things–New security and privacy challenges. Computer law & security review, 26(1), 23-30.

[4] Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. Computer Networks, 76, 146-164.

[5] Patton, M., Gross, E., Chinn, R., Forbis, S., Walker, L., & Chen, H. (2014, September). Uninvited connections: a study of vulnerable devices on the internet of things (IoT). In Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint (pp. 232-235). IEEE.

[6] Swan, M. (2012). Sensor mania! The internet of things, wearable computing, objective metrics, and the quantified self 2.0. Journal of Sensor and Actuator Networks, 1(3), 217-253.

[7] ForeScout, 2016, Know Your IoT Security Risk: How Hackable is Your Smart Enterprise? ForeScout IoT Enterprise Risk Report. Available on-line 16/9/2017. https://www.forescout.com/wp-content/uploads/2016/10/iot-enterprise-risk-report.pdf

[8] Singh, N. (2012). BYOD genie is out of the bottle–"Devil or angel". Journal of Business Management & Social Sciences Research, 1(3), 1-12.

[9] Koh, E. B., Oh, J., & Im, C. (2014). A study on security threats and dynamic access control technology for BYOD, smart-work environment. In Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol. 2, pp. 1-6).

[10] Bi, Z., Da Xu, L., & Wang, C. (2014). Internet of things for enterprise systems of modern manufacturing. IEEE Transactions on industrial informatics, 10(2), 1537-1546.

[11] Duncan, B., Happe, A., & Bratterud, A. (2016, December). Enterprise IoT security and scalability: how unikernels can

improve the status Quo. In Utility and Cloud Computing (UCC), 2016 IEEE/ACM 9th International Conference on (pp. 292-297). IEEE.

[12] Chang, S. I., Huang, A., Chang, L. M., & Liao, J. C. (2016). Risk factors of enterprise internal control: Governance refers to internet of things (iot) environment. RISK.

[13] Duncan, B., Whittington, M., & Chang, V. Enterprise Security and Privacy: Why Adding IoT and Big Data Makes It So Much More Difficult.

[14] Xiaohui, X., 2013, June. Study on security problems and key technologies of the internet of things. In Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on (pp. 407-410). IEEE.

[15] Myerson, J. M. (2002). Identifying enterprise network vulnerabilities. International Journal of Network Management, 12(3), 135-144.

[16] Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. Journal of Information Security and applications, 22, 113-122.

[17] Yu, Y., & Chiueh, T. C. (2004). Enterprise digital rights management: Solutions against information theft by insiders. Research Proficiency Examination (RPE) report TR-169, Department of Computer Science, Stony Brook University.

[18] Nurse, J. R., Erola, A., Agrafiotis, I., Goldsmith, M., & Creese, S. (2015, September). Smart insiders: exploring the threat from insiders using the internet-of-things. In Secure Internet of Things (SIoT), 2015 International Workshop on (pp. 5-14). IEEE.

[19] Siboni, S., Shabtai, A., Tippenhauer, N.O., Lee, J. and Elovici, Y., 2016. Advanced security testbed framework for wearable IoT devices. ACM Transactions on Internet Technology (TOIT), 16(4), p.26.

[20] Gordon Lyon. 2009. Nmap-Free security scanner for network exploration and security audits. https://nmap.org/.

[21] Aircrack-ng, a complete suite of tools to assess Wi-Fi network security. https://www.aircrack-ng.org/.

[22] Gerald Combs. 2007. Wireshark–A network protocol analyzer. https://www.wireshark.org/.

[23] Toh, J., Hatib, M., Porzecanski, O. and Elovici, Y., 2017, April. Cyber security patrol: detecting fake and vulnerable wifi-enabled printers. In Proceedings of the Symposium on Applied Computing (pp. 535-542). ACM.

[24] Homedale Wi-Fi/WLAN monitor tool, http://www.the-sz.com/products/homedale/.

[25] Morrow, B. (2012). BYOD security challenges: control and protect your most sensitive data. Network Security, 2012(12), 5-8.

[26] Miller, K. W., Voas, J., & Hurlburt, G. F. (2012). BYOD: Security and privacy considerations. It Professional, 14(5), 53-55.

[27] Nessus, a network vulnerability scanner, Tenable Network Security, http://www.tenable.com/products/nessus-vulnerability-scanner.

[28] Retina, a network security scanner, BeyondTrust, https://www.beyondtrust.com/products/retina-network-security-scanner/.

[29] NVD CVSS Calculator, https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator.

[30] Beyah, R., & Venkataraman, A. (2011). Rogue-access-point detection: Challenges, solutions, and future directions. IEEE Security & Privacy, 9(5), 56-61.

[31] Lewis, A., Li, Y., & Xie, M. (2016, October). Real time motion-based authentication for smartwatch. In Communications and Network Security (CNS), 2016 IEEE Conference on (pp. 380-381). IEEE.

[32] Migicovsky, A., Durumeric, Z., Ringenberg, J., & Halderman, J. A. (2014, March). Outsmarting proctors with smartwatches: A case study on wearable computing security. In International Conference on Financial Cryptography and Data Security (pp. 89-96). Springer, Berlin, Heidelberg.

[33] Denney, K., Uluagac, A. S., Akkaya, K., & Saputro, N. Demonstration of A Novel Storage Covert Channel on Android Smartwatch Using Status Bar Notifications.

[34] Wang, H., Lai, T. T. T., & Roy Choudhury, R. (2015, September). Mole: Motion leaks through smartwatch sensors. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (pp. 155-166). ACM.

[35] Maiti, A., Jadliwala, M., He, J., & Bilogrevic, I. (2015, September). (Smart) watch your taps: side-channel keystroke inference attacks using smartwatches. In Proceedings of the 2015 ACM International Symposium on Wearable Computers (pp. 27-30). ACM.

[36] Liu, X., Zhou, Z., Diao, W., Li, Z., & Zhang, K. (2015, October). When good becomes evil: Keystroke inference with smartwatch. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 1273-1285). ACM.

[37] Sarkisyan, A., Debbiny, R., & Nahapetian, A. (2015, November). WristSnoop: Smartphone PINs prediction using smartwatch motion sensors. In Information Forensics and Security (WIFS), 2015 IEEE International Workshop on (pp. 1-6). IEEE.

[38] Torre, I., Koceva, F., Sanchez, O. R., & Adorni, G. (2016, December). Fitness trackers and wearable devices: how to prevent inference risks? In Proceedings of the 11th EAI International Conference on Body Area Networks (pp. 125-131). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[39] Nahapetian, A. (2016, January). Side-channel attacks on mobile and wearable systems. In Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual (pp. 243-247). IEEE.

[40] Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., & Ribagorda, A. (2014). Evolution, detection and analysis of malware for smart devices. IEEE Communications Surveys & Tutorials, 16(2), 961-987.

[41] Liu, J., & Sun, W. (2016). Smart Attacks against Intelligent Wearables in People-Centric Internet of Things. IEEE Communications Magazine, 54(12), 44-49.

[42] Vanhoef, M., & Piessens, F. (2014, December). Advanced Wi-Fi attacks using commodity hardware. In Proceedings of the 30th Annual Computer Security Applications Conference (pp. 256-265). ACM.

[43] Mónica, D., & Ribeiro, C. (2011, September). Wifihop-mitigating the evil twin attack through multi-hop detection. In European Symposium on Research in Computer Security (pp. 21-39). Springer, Berlin, Heidelberg.

[44] Tokuyoshi, B. (2013). The security implications of BYOD. Network Security, 2013(4), 12-13.

[45] Eslahi, M., Naseri, M. V., Hashim, H., Tahir, N. M., & Saad, E. H. M. (2014, April). BYOD: Current state and security challenges. In Computer Applications and Industrial Electronics (ISCAIE), 2014 IEEE Symposium on (pp. 189-192). IEEE.

[46] Rivera, D., George, G., Peter, P., Muralidharan, S., & Khanum, S. (2013). Analysis of security controls for BYOD (bring your own device).

[47] Costantino, G., Martinelli, F., Saracino, A., & Sgandurra, D. (2013, December). Towards enforcing on-the-fly policies in BYOD environments. In Information Assurance and Security (IAS), 2013 9th International Conference on (pp. 61-65). IEEE.

[48] Tanimoto, S., Yamada, S., Iwashita, M., Kobayashi, T., Sato, H., & Kanai, A. (2016, October). Risk assessment of BYOD: bring your own device. In Consumer Electronics, 2016 IEEE 5th Global Conference on (pp. 1-4). IEEE.

[49] Ratchford, M., Wang, P., & Sbeit, R. O. (2018). BYOD Security Risks and Mitigations. In Information Technology-New Generations (pp. 193-197). Springer, Cham.

[50] Uma, M., & Padmavathi, G. (2013). A Survey on Various Cyber Attacks and their Classification. IJ Network Security, 15(5), 390-396.

[51] Chen, C. M., Yang, P. Y., Ou, Y. H., & Hsiao, H. W. (2014, May). Targeted Attack Prevention at Early Stage. In Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on (pp. 866-870). IEEE.

[52] Le, T. M., Liu, R. P., & Hedley, M. (2012, September). Rogue access point detection and localization. In Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on (pp. 2489-2493). IEEE.

[53] Anmulwar, S., Srivastava, S., Mahajan, S. P., Gupta, A. K., & Kumar, V. (2014, February). Rogue access point detection methods: A review. In Information Communication and Embedded Systems (ICICES), 2014 International Conference on (pp. 1-6). IEEE.

[54] Conti, M., Delmastro, F., Minutiello, G., & Paris, R. (2013, November). Experimenting opportunistic networks with WiFi Direct. In Wireless Days (WD), 2013 IFIP (pp. 1-6). IEEE

[55] SmartGeekWrist, 11 Best Standalone Smartwatch With Sim Card You Need to Check. Available on-line 23/9/2017. https://www.smartgeekwrist.com/standalone-smartwatch-sim-card/.

ABOUT THE AUTHORS:

Shachar Siboni is a Ph.D. student in the department of software and information systems engineering (SISE) at Ben-Gurion University of the Negev (BGU). Shachar holds B.Sc. and M.Sc. degrees in communication systems engineering (CSE) from BGU as well. His research interests include security risk analysis and machine learning approaches in the Internet of Things (IoT) research domain. For the last 15 years, Shachar has worked in a variety of roles at leading companies in the hi-tech industry, ranging from IT technical leader, communication systems engineer, real-time embedded software engineer/developer and team leader, to security researcher and project manager. In his most recent role, Shachar led a joint research project, collaborating with research groups from BGU's Cyber Security Research Center (CSRC) and the iTrust Centre for Research in Cyber Security at Singapore University of Technology and Design (SUTD).

Asaf Shabtai is an assistant professor at the Department of Software and Information Systems Engineering at Ben-Gurion University. Asaf is a recognized expert in information systems security and has led several large-scale projects and researches in this field. His main areas of interest are computer and network security, machine learning, cyber intelligence, security awareness, security of IoT and smart mobile devices, social network analysis, and security of avionic and operational technologies (OT) systems. Asaf holds a B.Sc. in Mathematics and Computer Sciences; B.Sc. in Information Systems Engineering; M.Sc. in Information Systems Engineering and a Ph.D. in Information Systems Engineering all from Ben-Gurion University. Since 2005, Asaf has been a principle investigator of various research projects funded by Deutsche Telekom AG, (Telekom Innovation Laboratories @ BGU), Israeli Ministry of Defense, Israeli Ministry of Trade and Commerce, and several leading cyber security companies. Asaf has published papers in leading peer-reviewed journals and conferences. In addition, he has co-authored a book on information leakage detection and prevention.

Yuval Elovici is the director of the Telekom Innovation Laboratories at Ben-Gurion University of the Negev (BGU), head of BGU Cyber Security Research Center, and a Professor in the Department of Software and Information Systems Engineering at BGU. He holds B.Sc. and M.Sc. degrees in Computer and Electrical Engineering from BGU and a Ph.D. in Information Systems from Tel-Aviv University. For the past 14 years he has led the cooperation between BGU and Deutsche Telekom. Prof. Elovici has published articles in leading peer-reviewed journals and in various peer-reviewed conferences. In addition, he has co-authored a book on social network security and a book on information leakage detection and prevention. His primary research interests are computer and network security, cyber security, web intelligence, information warfare, social network analysis, and machine learning. Prof. Elovici also consults professionally in the area of cyber security and is the co-founder of Morphisec, startup company that develop innovative cyber-security mechanisms that relate to moving target defense.

Selecting Suitable Virtual Machine Migrations for Optimal Provisioning of Virtual Data Centers

M. P. Gilesh
Government Engineering College Wayanad
Kerala, India - 670644
gileshmp@gecwyd.ac.in

Sanjay Satheesh
National Institute of Technology Calicut
Kerala, India - 673601
meetsanjaysatheesh@gmail.com

S. D. Madhu Kumar
National Institute of Technology Calicut
Kerala, India - 673601
madhu@nitc.ac.in

Lillykutty Jacob
National Institute of Technology Calicut
Kerala, India - 673601
lilly@nitc.ac.in

ABSTRACT

Maximizing the number of virtual infrastructures spawned out of a data center is a prime concern of cloud service providers to improve their revenue and the customers' quality-of-experience. Optimal placement of topology sensitive virtual data centers on its physical counterparts increases the resource utilization and thereby the revenue. However, provisioning, scaling, and de-provisioning of virtual data centers, over the time, leaves the cloud data center fragmented. Embedding new virtual data centers on a fragmented data center would require migration of virtual machines and virtual network, already mapped to it. However, these migrations are costly in terms of the resource usage and possible violations of the terms of service level agreement. The costs of these migrations associated with a new embedding should be controlled to increase the profit. In this paper, the problem of finding virtual machine migrations with minimum cost is formally defined and two meta-heuristic solutions are proposed for it. We consider the memory size, the rate of page dirtying, the class of application running on the VM, and the bandwidth of the migration path for modeling the cost of migration. The objective is to find a pattern of virtual machine migrations so that the total cost for embedding a new virtual data center is minimized. Experimental results show that the proposed technique can reduce the average migration time upto 30% and the penalty for corresponding service level agreement violation by 200%.

CCS Concepts

•Computer systems organization → Cloud computing; •Networks → Cloud computing;

Keywords

Cloud Computing, Virtual Machine Migration, Resource Allocation, Virtual Data Center, Artificial Bee Colony Optimization, Simulated Annealing.

Copyright is held by the authors. This work is based on an earlier work: SAC'18 Proceedings of the 2018 ACM Symposium on Applied Computing, Copyright 2018 ACM 978-1-4503-5191-1. http://dx.doi.org/10.1145/3167132.3167153.

1. INTRODUCTION

During the last decade, there has been a paradigm shift in the way enterprises, and the retail users deploy their applications. Leasing resources from the cloud has proven to reduce the cost as well as the time to roll-out applications and increase the flexibility of such applications. Higher demand for cloud-based application deployment increases the probability of rejection resulting from the deficit of resources in the cloud data center (CDC). Virtual data center (VDC), a set of virtual machines(VMs) connected by a particular topology, has become a prevalent model of service provisioning in the cloud-centric business environment. The problem of embedding VDCs on a CDC to meet the customer and cloud service provider (CSP) objectives is called virtual data center embedding (VDCE). The optimal VDCE is hard to solve because of the individual demands for virtual machines along with the bandwidth and delay constraints of the interweaving topology [3]. If free resources are not available in the structure demanded by the VDCs, migrations would be necessary to satisfy the request. For example, the runtime status of a *pod* of fat-tree [1] data center network is shown in Figure 1. There are two VDCs already embedded in the CDC. Servers S2 and S3 are already saturated. The incoming VDC request with one Type 1 VM and two Type 2 VMs (differentiated by the width of the rectangle) with a common *vSwitch* need to be embedded on the CDC. Assume that the Type 1 VMs on servers 2 and 3 consumes ten units of bandwidth. The incoming VDC cannot be embedded directly on the CDC because there is a shortage of link bandwidth to server S1, though there is enough space for the Type 1 VM. However, collective resources of all the components of the CDC is sufficient to provision the new VDC. The VDC can be embedded if one of the VM embedded on server S2 is migrated to server S1.

Existing works in the VDCE, either do not fully leverage migrations during the process of embedding or they use periodic consolidation to create room for future embedding, or scaling requests [21]. The distribution of residual computing and network resources in the CDC may result in unfulfilled resource requirements for embedding a new VDC request or auto-scaling the incumbent applications. In real scenarios, the resource availability in different Availability Zones (AZs) may also be a constraint in accommodating the future

resource demands. Overlooking the possibility of VM migrations to create space for expected resource demands can result in a poor rate of VDC acceptance and hence, is detrimental to the revenue of the CSP. Periodic consolidations of resources, by migrating VMs, help to overcome this fragmentation at the cost of performance and acceptance rate. A CSP would like to increase his profit while embedding a new VDC, by allowing only the inexpensive VM migrations.

This paper addresses the problem of finding the cost-effective VM migrations, if any, to optimally embed a new VDC request across the fragmented resources. Unlike the existing work in the literature [4] [21] [20], two conflicting requirements are considered simultaneously - minimizing the migration cost and optimized embedding of a VDC. Employing the least cost migrations [21] may not always end up in a feasible solution. The challenge is to find a cost-effective set of VM migrations to embed the VDC. The minimization problem, viz. the *MinCostMigration* problem, finds a set of VM migrations with a minimal cost that embeds a new VDC on the CDC. The problem is NP-hard and difficult to solve. Inquiring whether there exists a migration pattern with the cost of all migrations less than C_{max} is the decision version of the problem, viz. the *BoundedCostMigration* problem. We use the duration of migration, which is the common interpretation of the cost of migration, as the cost for the purpose of analysis. The total migration time of a VM is determined by its memory size, the rate of memory page dirtying, length of the migration path (latency), the bandwidth of the bottleneck link and the class of the application(s) hosted by the VM [16] [13]. Empirical results show the cost-benefit from controlled VM migrations during resource allocation. Important contributions of this paper are:

- The formal definition of the problem of finding cost-effective and bounded-cost VM migration pattern to maximize the resource utilization in cloud data centers.
- Two different meta-heuristic solutions for the problem with empirical evaluation in the context of VDC embedding.

The rest of this paper is organized as follows. Section 2 gives an outline of the existing work in migration aware VDC embedding. We describe the formulation of both *MinCostMigration* problem and *BoundedCostMigration* problem in Section 3. In Section 4, we give detailed explanations of a heuristic and two meta-heuristic algorithms for solving the problem. We give the results of experimental evaluation of our solutions in Section 5. Section 6 concludes the paper.

2. RELATED WORK

Virtual data center embedding algorithms solve the challenge of meshing multiple VMs along with interconnecting bandwidth or a virtual network onto the CDC. Hardly any existing algorithms exploit the possibility of VM migrations for the optimal embedding of multiple VDCs. Most of the existing VDCE algorithms [2] [8] [19] [5] consider edge virtualization i.e., VMs with interconnecting bandwidths wherein the service providers have hardly any flexibility in network

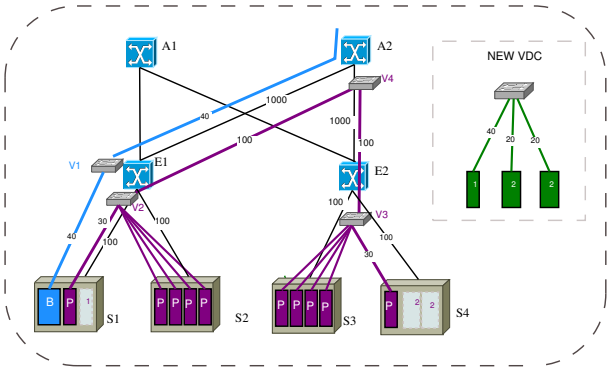


Figure 1: Motivating example for using VM migration for optimal VDC provisioning

management. The VDCE problem in such an environment reduces to the VM placement problem with bandwidth specification and does not consider embedding of any interconnecting topology. The existing VDCE algorithms in full virtualization aim at maximizing revenue, increasing acceptance, checking energy consumption, increasing reliability, or load balancing. Mixed integer programming (MIP) / Integer linear programming (ILP) solutions yield higher acceptance at the cost of longer time to embed. Heuristic algorithms solve the VDCE problem faster, compromising the optimality. Zhani et al. [21] exploit the possibility of VM migrations to increase the profit from VDC embedding by a greedy selection of possibly inactive VMs migration and by a periodic consolidation of VMs. However, as shown later, greedy selection of migrations often fails to find an embedding solution, though there exists one. Moreover, the periodic consolidation will be beneficial only if the scheduling of migrations are done accurately based on the rate of application activity. Thus, the above techniques rely on periodic consolidation of CDC resources to de-fragment the data center to serve future requests.

To the best of our knowledge, only CQNCR [4] aims at finding an optimal VM migration pattern for attaining a desired final configuration of VM placements, starting from an initial configuration. The CQNCR tries to minimize the time required for completing all the migrations of the pattern to reach the given final shape. The technique gives an optimum solution when there exists only one final configuration that solves the embedding problem. However, CQNCR neglects the fact that there can be multiple final configurations which satisfy the resource allocation requirement. Moreover, the authors do not consider the influence of network flows from co-existing VMs and other factors that count for the duration of migration. An essential drawback of the CQNCR is the excessive use of high bandwidth links for faster movement, resulting in the disruption of services of other tenants.

The problem addressed in this paper is different from those in the literature, because it tries to find the best set of VM migrations with minimum or no violation of SLAs to enable an incoming VDC to get embedded on the CDC, satisfying the constraints. To best of our knowledge there exist no work which formalizes the problem to meet a particular resource requirement.

3. PROBLEM FORMULATION

A cloud data center is represented as an undirected graph G^s with switches (N^s) as interior nodes, hosts/servers (M^s) as terminal nodes, and the connecting cables/links (E^s) as edges. Similarly, G^v represents a virtual data center request. The VDCs get provisioned, scaled, and de-provisioned from the system dynamically based on the availability of resources. In any interval of time, there can be VDCs arriving and leaving the CDC. At any point of time only one VDC is considered for embedding onto the CDC. Variable $\Re'$ represents the set of all the VDCs currently mapped to the CDC. Repeated provisioning, scaling, and de-provisioning of VDCs, over time, may create fragmentation of CDC resources. It is hard to find a VDC mapping across the fragmented resources, though there exists sufficient resources to meet the VDC requirements. Thus, fragmentation necessitates consolidation of resources to satisfy the demands of a new VDC. Migration of VMs becomes essential at this point to unveil the resources to embed the new VDC. However, choosing any set of fragmented resources may not be a good choice in terms of the cost of required VM migrations. There could be exponentially many combinations of VM migrations that create room for the incoming VDC. The problem is to find an optimal set of VM migrations, in terms of the cost, such that the incoming VDC u can be embedded. We define the optimization and the decision versions of this problem with the following assumptions.

1. The incoming VDC request can be embedded with zero migration cost if there are sufficient de-fragmented resources or with positive migration cost if at least one of the existing VMs need to be migrated to create space for new VDC.

2. The links and switches of the data center have sufficient capacities to embed their virtual counterparts.

3. The cost of provisioning a new VM, excluding the migration cost, is constant and same across the servers.

3.1 The MinCostMigration problem

On embedding a VDC request, a CSP is primarily concerned about increasing the profit from that embedding. The gain can be maximized by increasing the revenue (usage charges from customers), minimizing the embedding and maintenance costs, or both. The CSP can exploit the possibility of VM migrations to increase the number of VDC embeddings (acceptance rate) and thus the revenue. However, the cost of such migrations should not deceive the revenue from embedding the VDC. The objective of this problem is to find the least cost migration pattern so that the static cost of deploying the new VDC is minimum. The cost of migration is defined in terms of the size of VM, page dirtying rate of the VM, and the bandwidth of the migration path [13]. Size of a VM may include the size of disk in case of directly attached storage configurations. The VMs run applications belonging to various classes which will determine the page dirtying rate. The bandwidth of a migration path is that of the bottleneck link. Higher the available bandwidth, lower would be the cost of migration.

Table 1: Variables used

Var	Description
$\Re$'	The set of VDCs embedded in CDC
G_t^v	The new VDC to be embedded
N^s	Set of CDC switches
E^s	Set of CDC links
M^s	Set of servers in CDC
$\mathcal{P}^s$	Set of all paths in CDC network
N_u^v	Set of virtual switches of VDC u
E_u^v	Set of virtual links of VDC u
M_u^v	Set of VMs of the VDC u
ϕ_a^s/ϕ_a^v	Attribute a of server (in CDC) / VM (of a VDC)
δ_b^s/δ_b^v	Attribute b of switch of CDC / VDC
ξ_c^s/ξ_c^v	Attribute c of link of CDC / VDC
X	1 if the new VDC is embedded; 0 Otherwise
X_u	1 if the VDC u remains embedded; 0 Otherwise
P	Revenue from embedding new VDC G_t^v
$size$	The RAM size of the VM (Bytes)
R_i	Number of pages dirtied in i^{th} iteration
B_{avg}	Avg. bandwidth of the migration path (bps)
tm_i^u	Max migration time allowed by SLA for VM i
num	Total number of pages in the RAM
p_size	The size of a memory page (Bytes)

Formally, an undirected graph $G^s = (M^s, N^s, E^s, \phi_a^s, \delta_b^s, \xi_c^s)$ represents the CDC, where ϕ_a^s, δ_b^s, and ξ_c^s are the server, switch, and link capacities of the CDC, respectively. Similarly, a VDC is represented as $G^v = (M^v, N^v, E^v, \phi_a^v, \delta_b^v, \xi_c^v)$ wherein M^v, N^v, and E^v are the set of VMs, virtual switches, and virtual links and ϕ_a^v, δ_b^v, and ξ_c^v are the respective capacities of the features under consideration. $\Re'$ is the set of VDCs currently embedded and this mapping is represented as

$$\mathcal{F} : (M^v, N^v, E^v \rightarrow (M^s, N^s, \mathcal{P}^s)$$

$\forall G^v \in \Re'$ where, $\mathcal{P}^s$ is the set of simple paths in the CDC. A new VDC G_t^v needs to be embedded on the CDC G^s. The objective of the *MinCostMigration* problem is to minimize

$$\sum_{u \in \Re'} \sum_{i \in M_u^v} \sum_{l \in \{M^s \setminus \mathcal{F}(i)\}} C_{il}^u . X_u - X.P \tag{1}$$

such that a mapping $\mathcal{F}'$ is found for every $G^v \in (\Re' \cup G_t^v)$. The variable $X \in \{0, 1\}$ indicates whether the VDC was embedded or not. P is the revenue from embedding the new VDC. C_{il}^v is the cost of migrating a VM i to server l and should be interpreted as the expected penalty for violation of service level agreement conditions. The problem is a constrained optimization problem wherein all the incumbent VDCs remain embedded in the CDC i.e., $X_u = 1$, $\forall u \in \Re'$. Moreover, the following constraint embeds the VDC G_t^v if the capacity constraints of the CDC are met.

$$X \in \{0, 1\} \tag{2}$$

The cost of migration C_{il}^u can be expressed as,

$$C_{il}^u = max\left(0, \frac{8(size + \sum_{i=1}^{n}(num \times R_i) \times p_size)}{B_{avg}} - tm_i^u\right)$$

where $size$ is the RAM size of the VM and p_size is the size of each page. R_i is the number of pages dirtied in i^{th}

iteration of the migration process and B_{avg} is the average bandwidth of the migration path and t_m the maximum permissible duration for migration as per the SLA. The total number of iterations of a pre-copy migration [6] instance is denoted by n. The value of num is calculated as

$$num = \frac{size}{p_size}$$

Other constraints, same as that of VDCE [7], make sure that if the new VDC is embedded on the CDC after migrating some VMs, the capacities of the servers, switches and links of the CDC are not violated.

If the embedding of the new incoming VDC can be accomplished without migrations, the cost of migrations is 0. However, the problem of embedding the incoming VDC request, even without using migrations, remains NP-Hard.

THEOREM 1. *MinCostMigration problem is NP-Hard.*

PROOF. We prove the hardness of MinCostMigration problem by reducing the feasibility version of the Multi Knapsack Problem (MKP) [12]. Let $K = (N, M)$ be a given instance of MKP where N is the set of n items and M is the set of m sacks. Each $n_i \in N$ has weight of w_i and constant profit p, and each sack $m_j \in M$ has a capacity c_j. Construct a VDC instance $G^v = (M^v, N^v, E^v, \phi^v_a, \delta^v_b, \xi^v_c)$ and a CDC instance $G^s = (M^s, N^s, E^s, \phi^s_a, \delta^s_b, \xi^s_c)$ such that

- M^s denotes the set of sacks in M (all sacks in M is assumed to be empty i.e., all the incumbent VDCs are unmapped i.e., liable to change its current mapping).
- Each VM in M^v denotes item n_i in N
- Each vertex v_j in M^s is associated with a capacity $\phi^s(v_j)$ which denotes the capacity C_j of the sack m_j in M.
- Each vertex v_i in M^v is associated with a size $\phi^v(v_i)$ which denotes the weight W_i of the items n_i in N.

CLAIM 1. *If there is a mapping from the VDC to CDC such that the capacity constraints of v_j in M^s are not violated, then there is a solution for the given MKP problem.*

PROOF. Suppose $V_i \subseteq M^v$ is mapped to the node $v_j \in M^s$ such that $\sum_{v_i \in V_i} w_i = c_j$ then the subset $N_i \subseteq N$ is assigned to the sack $m_j \in M$.

CLAIM 2. *If there is an optimal solution for the MKP then there exists an optimal mapping from VDC to CDC. If the subset N_i of N is assigned to the sack $m_j \in M$ then $V_i \subseteq M^v$ can be mapped to the node v_i of M^v where V_i contains the set of nodes corresponding to the items in the subset N_i*

PROOF. Similar to that of the Claim 1.

Hence, it is proved that the MinCostMigration problem is NP-hard.

3.2 The BoundedCostMigration problem

This is the decision version of the *MinCostMigration* problem and can be useful in embedding a request only if the cost of migrations involved is within a permissible limit. If C_{max} is the maximum cost of migration affordable while entertaining an incoming VDC then it is non-profitable to embed the new VDC if the penalty associated with the SLA violations exceed C_{max}. If there exists a set of migrations with the cost less than C_max, the new VDC can be embedded. Hence, a C_{max}-*BoundedCostMigration* problem is to find a pattern of VM migrations that would allow the embedding of a new incoming request and the total penal cost associated with these migrations does not exceed C_{max}. The problem is NP-complete and the challenge is to check whether there exists a pattern of VM migrations whose total cost does not exceed C_{max}. The capacity constraints of the CDC must be satisfied by the above embedding, inclusive of the requirements of new VDC.

Both the problems described above are different and innovative compared to the existing works in the literature by Bari et al.[4] and Zhani et al. [21] because our problems encompass the objective of a feasible virtual data center embedding and an optimal set of VM migrations, in terms of the penalty for resulting SLA violations, to accomplish the embedding task. We assume the conditions of the SLA similar to that of VMWare vCloud Air [18]. Similar to the model [14], a consumer requests virtual computing and network entities (VMs and VN) of a VDC along with the service level specifications. The resource manager of the data center calculates the feasibility of embedding, including the cost of expected migrations. An SLA credit is earned by a consumer when the CSP violates (for a qualifying duration) the guarantee on the quality of services. The qualifying duration of a violation and credit value are determined based on the type of service offered.

4. PROPOSED SOLUTIONS

The primary objective of the solutions proposed here is to find a set of migrations with minimum time across all the VMs being relocated whilst finding a feasible embedding for a new VDC. Being a combinatorial optimization problem, the simplest solution would be to apply heuristics to select the VM migrations in the ascending order of their expected time. The migrations are done until there is sufficient de-fragmented resource to embed the incoming request. We also propose to use a swarm intelligence based meta-heuristic algorithm called Artificial Bee Colony(ABC) optimization [9]. Further, a Simulated Annealing [11] meta-heuristic algorithm is proposed to find the solutions faster with the backing of a perturbation function. The available bandwidth is a constant (reserved for migration) during the migration window, but different for different VMs.

4.1 Heuristic Algorithm

The heuristic algorithm attempts to find a pattern of minimum cost migrations. The idea is to sort the VMs in terms of cost of migrations i.e., in the increasing order of the expected time for migration. For each VM i in the list, find VMs in other servers,say j, such that the sum of residual resources of server j and the difference between the size of

i with a VM k $(\in j)$ is equal to the size of a VM l in the new VDC. That is, the VM i is migrated to server j, if the following equation is satisfied for some k, l

$$s_k^{u1} - s_i^{u2} + residual(j) = s_l^{u3} \tag{3}$$

where s_k^{u1} is the size of VM k belonging to VDC $u1$, s_i^{u2} is the size of VM i belonging to VDC $u2$, $residual(j)$ is the available CPU resources of server j, and s_l^{u3} is the size of a VM in VDC $u3$. The heuristic method is formally given in Algorithm 1.

The heuristic algorithm runs fast but does fail, with high probability, in finding a solution to the problem, if the residual resource of the CDC is just enough to embed the VDC. Moreover, the sum of costs of the associated VM migrations is suboptimal. Artificial bee colony meta-heuristic and simulated annealing are used to overcome the limitations of the heuristic algorithm.

Algorithm 1: Heuristic MinCostMigration

```
Input: CDC G^s, Current mapping F, New VDC G^v.
Output: New mapping F .
Function Heuristic_MinMigCost(G^s, F, G^v):
    t_list = { }                // List of incumbent VMs
    for VM i mapped to G^s do
        d_i^u = page dirtying rate
        b_i^u = Available band width
        s_i^u = Memory size of VM i
        Calculate tm_i^u    // migration time of i [13]
        t_list.add(i, tm_i^u)
    end
    Sort t_list in ascending order of tm_i^u.
    repeat
        for i ∈ t_list do
            while F(i) ≠ j do
                if ∃ VM k ∈ j : Eq. (3) is satisfied then
                    Migrate i to j
                    Migrate k to F(i)
                    Embed new VM (∈ G^v) on j
                else
                    Continue
                end
            end
        end
    until ∃ VM (∈ G^v) to embed
end
```

4.2 Artificial Bee Colony (ABC) Optimization

Swarm Intelligence represents the branch of the artificial intelligence that investigates individuals' actions in different decentralized systems. These decentralized systems are composed of entities that communicate, cooperate, collaborate, exchange information and knowledge, and perform some tasks in their environment. Bees in nature look for food by exploring the fields in the neighborhood of their hive. They collect and accumulate the food for the future use of all bees in the hive. The collective behavior of bees is simulated in the ABC optimization technique.

Algorithm 2: ABC MinCostMigration

```
Input: CDC G^s, Current mapping F, New VDC G^v .
Output: New mapping F .
Function ABC_MinMigCost(G^s, F, G^v):
    iter = 0 Initialize the population P
    Calculate the f_i, ∀i ∈ P
    Best Migration Cost ( MC_b)← ∞
    while iter ≤ MAX_ITER or MC_b > 0 do
        for employed bee f_i do
            i ← New solution with eq (5).
            Calculate f_i = fitness of i.
            Apply Selection
        end
        Calculate probability p_i using eq (4)
        for onlooker bees do
            j ← New solution with eq (5).
            Calculate f_j = fitness of j.
            Apply Selection
        end
        if ∃ abandoned solution then
            Assign new random solution to scouts.
        end
        if New_Cost < MC_b then
            MC_b ← New_Cost
        end
        iter = iter + 1
    end
end
```

In ABC algorithm [9] [10], the colony of artificial bees consists of three types of bees: *employed bees, onlookers, and scouts*. One half of the colony is the employed bees and the other half is the onlookers. The number of employed bees is equal to the number of food sources around the hive [17]. Initially, the ABC generates a randomly distributed initial population P of N solutions i.e., food source positions, where N denotes the size of the population. The position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. After the initialization step, the solution is subjected to repeated cycles of changes based on the search results of the employed bees, the onlooker bees, and scout bees. An employed bee modifies the position (solution) in her memory only if the fitness of the new source (new solution) is better than the old. After all employed bees complete the search process for the food sources, they share the information of the food sources and their position with the onlooker bees through a *waggle dance*. An onlooker bee evaluates the information shared by the employed bees and chooses a food source with a probability related to its quality. An onlooker bee chooses a solution depending on the probability value associated with that solution, p_i, as the relative quality of the solution i with respect to other solutions, calculated as

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{4}$$

where f_i is the fitness of solution i.

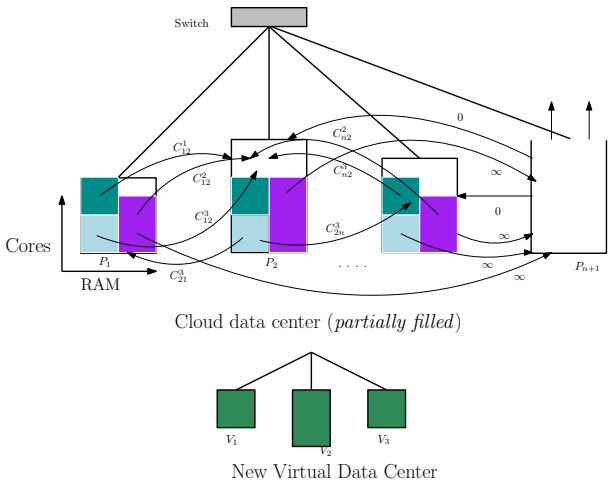


Figure 2: The configuration for solving the MinCostMigration and BoundedCostMigration problems

New candidate solutions are created from the old solutions using the expression

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{5}$$

where x_{ij} and x_{kj} are the jth parameter of the ith and k^{th} solutions respectively. Values of k and j are random within the range of number of solutions and the dimension of solutions, respectively. ϕ_{ij} is a random number between [-1, 1] to form a nearby solution point. The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In ABC, this is simulated by producing a position randomly and replacing it with the abandoned one. The repeated search process continues until the optimum is found or the maximum number of iterations is reached.

4.2.1 ABC for MinCostMigration

The layout shown in Figure 2 is used to apply the ABC optimization to *MinCostMigration* problem. There are n servers and has capacities represented along the axes of CPU cores and RAM of the corresponding server (see Figure 2). Further, an $(n+1)^{th}$ hypothetical *infinity* server has an infinite capacity (or the capacity to accommodate all VMs including VMs of the new VDC).

As an initial solution, the VMs of the new VDC get placed into the *infinity* server. The cost of migrating a VM from the real server to the *infinity* server is set to ∞ and zero from *infinity* server to others. All the initial solutions are formed by migrating VMs among the servers and from the *infinity* server to real servers. When the foraging bees look for solutions in subsequent iterations, the solution points using migrations to the *infinity* server will be automatically discarded due to its high cost. In each iteration a new solution point is created by migrating a VM from one server to another. To ensure that the topology is also maintained, the link and switch constraints of the incumbent VDCs also are considered. The formal artificial bee colony optimization for the *MinCostMigration* problem is given in Algorithm 2. The algorithm iterates for a predetermined number of rounds or until a solution with 0 migration cost (the ideal case) is found. If the algorithm terminates after the maximum number of rounds, MC_b is taken as the cost and corresponding migration pattern as the solution. The algorithm may terminate without a solution if it halts with a solution with VMs in the *infinity* server. In the case of the bounded cost version of the problem, the algorithm stops after the iteration that finds a migration pattern with MC_b less than C_{max}.

The ABC optimization is seen to converge slower than expected due to the excessive search of local regions by the onlooker bees. Most of such searches end up in the abandonment of the solution (the food source) due to the complexity associated with topology matching. Further, it is also noticed that the random perturbation (equation (5)) is not effective enough for solving *MinCostMigration* problem. As an alternative, simulated annealing assisted by a perturbation function is used to find the solution faster. Later, it is shown that simulated annealing finds solutions which ABC algorithm fails to.

4.3 Simulated Annealing Method

The proposed algorithm is basically derived from *Simulated Annealing* meta-heuristic, which is appropriate for combinatorial optimization problems [11]. Simulated annealing is capable of finding a good approximation to the global optimum of a given function in a large search space. It has a strong local search ability which prevents from getting stuck at local optimal solutions in the process of searching. Simulated annealing was inspired from nature: annealing, in metallurgy and materials science, is a heat treatment that alters the physical and sometimes chemical properties of a material.

Simulated Annealing starts with an initialization phase in which an initial feasible solution for the problem is constructed and an initial value is set. Then, several iterations are performed and different solutions are explored. After each iteration, the value of the temperature is updated. When this value reaches a threshold, the algorithm ends with an optimal sequence of migrations. During each iteration, a *perturbation* function is called which returns a desirable solution i.e., a better migration pattern. Each of this solution is accepted either unconditionally, if the solution improves the objective function value, or with a certain probability, which is a function of temperature and difference of the two objective function values, otherwise. Accuracy of the solution depends on the quality of the corresponding move and the cooling schedule, which includes the initial and final temperature values, the number of iterations, and the temperature decreasing law.

The algorithm takes as input parameters, the current allocation of VMs, the incoming VDC request, the list of VMs and PMs with their specifications, initial and final temperature value and a cooling factor for the temperature decreasing law. The goal is to find the set of VM migrations that minimize the cost incurred for migration, expressed in (1). The proposed algorithm uses the current allocation, with the new VDC request embedded into hypothetical infinity server which has infinite resources, as the initial solution. The initial value of the objective function is set as infinity. The two main concepts of our algorithm, the *perturbation*

function and the criterion for accepting a new solution are explained in the following sections.

4.3.1 Perturbation Function

This function produces a new migration pattern as a possible current solution. The new migration pattern is obtained by perturbing on a VM i.e., by migrating a single VM from a source node to destination node. In order to make the algorithm converge faster to a solution, the migration pattern is not selected randomly. Instead, a *desirability* factor is introduced, based on which the migration pattern is selected. This function takes the current allocation of VMs on the physical nodes, the incoming VDC request, the list of VMs and PMs with their specifications and a single VM (which is selected as source) as input parameters. The desirability value of the source VM, received from simulated annealing algorithm, is compared with a threshold. The desirability of the source is inversely proportional to the utilization of the source node. Once the source node satisfies the source desirability check i.e., if it is greater than the threshold, the VM needs to be migrated to another physical machine.

Algorithm 3: Perturbation

```
Input: CDC G^s, Current mappings F, New VDC G^v,
       A source VM i ∈ VM_list.
Output: mig_list
Function Perturbation((G^s, F, G^v, i)):
    Get source_node
    if desirability(source_node ≥ src_threshold then
        for server j do
            if source_node ≠ j && migration of VM i
              to j is possible then
                Append migration to mig_list.
                return mig_list
            end
        end
        destination= DesiredDest(G^s, F, i)
        Append migration to mig_list.
        Perturbation(G^s, F, i, destination.VM)
    end
end
```

If there is another physical machine which has enough space to accommodate the source VM, the migration is added to *mig_list*. If there is not enough space in any of the servers, space for the source VM should be made available by migrating one or more VM(s) from other physical nodes. In that case, the desirability of the destination node should be computed. The *destination desirability* function returns a server which is the best possible destination node for that particular VM along with the best VM in that destination node, that needs to be migrated to accommodate the source VM. The destination node is selected based on the change in resource utilization for that node. The VM that needs to be migrated from the destination node is selected based on the size of VM. The VM that uses lowest possible resources in the destination node is selected for migration. The migration process from the source node to destination node is added to *mig_list*. Recursively the *perturbation* function is called with the selected VM from the destination node as the source VM. The formal description of the perturbation function is shown in Algorithm 3.

4.3.2 Simulated Annealing Algorithm

The VM allocation is updated according to the migration which was selected through the perturbation function. The objective function is calculated using the set of migrations from the *perturbation* function. If the objective function value of the new solution *new_sol* is better than the current solution *curr_sol*, the *new_sol* is set as *curr_sol*. Otherwise, if the *new_sol* is not better than *curr_sol*, still the *new_sol* may be set as the *curr_sol* based on an acceptance probability. After each iteration the value of initial temperature is updated. The formal description of the proposed algorithm is given in Algorithm 4.

Algorithm 4: Simulated Annealing

```
Input: CDC G^s, Current mappings F, New VDC G^v,
       init_temp, cooling, acc_threshold.
Output: New mapping F.
Function SimAnnealing((G^s, F, G^v, T, C, A)):
    Set curr_obj using the equation (1)
    curr_temp = init_temp
    while acc_prob ≥ A do
        for VM i do
            mig = Perturbation(G^s, F, G^v, i)
            Update curr_map
            Compute the new_obj using the equation (1).
            δ = new_obj − curr_obj
            if δ ≤ 0 then
                accept = True
            else
                Compute the acceptance probability.
                if The migration is accepted by the
                  probability then
                    accept = True
                end
            end
            if accept = True then
                curr_obj = new_obj
            else
                Undo the migration
                Update the curr_map.
            end
        end
    end
end
```

5. SIMULATION AND RESULTS

The NetworkX library of Python and FNSS [15] are used to manage the CDC and VDCs in the simulation environment. The physical data center uses three popular topologies used in real data centers, viz. Three-tier Clos, Fat Tree and Jelly Fish. The preliminary experiments use star topology in VDCs, to avoid the need of embedding complex virtual network of the VDC. Ten test cases with different topologies and characteristics are used in the evaluation. The empirical evaluation setup has CDCs with 3 to 6 physical machines,

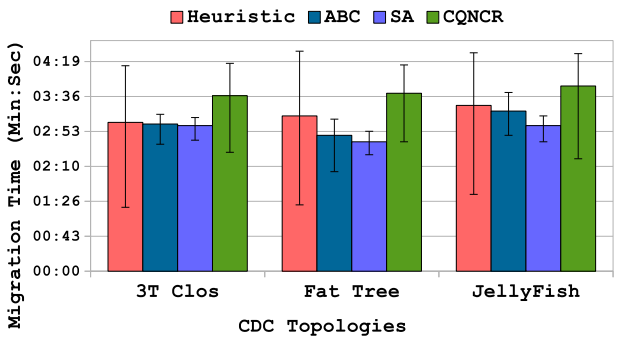


Figure 3: Comparison of the average migration time for the heuristic, ABC, SA, and existing method.

each configured with 8-16 CPU cores and 32-128 GB RAM. The VDCs have 4 to 100 virtual machines, each configured with 1-8 CPU cores and 2-16 GB RAM. The virtual and physical link requirements are abstracted by the bandwidth availability between the VMs. The new request consists of 1 to 50 virtual machines with the above VM configuration, connected over a star topology. The storage is assumed to be shared and hence doesn't count to the migration time. The migration costs are assigned to the VMs based on the type of application - network intensive, CPU intensive and balanced. The applications are uniformly distributed across the VDCs. Migration times are different for various CDC topologies. There are 10 different test-cases, each with different behavior. Major challenge is to create test-cases in which modifications are required to embed new requests. When the data center is large, creating data sets with a challenging initial configuration is a hard task.

Performance of the proposed algorithms is compared against the CQNCR[4]. CQNCR assumes a final configuration to the VDC embeddings before finding the suitable migration pattern. Existing VDCE batch embedding [7] algorithm is used to find a solution to VDCE to set the final configuration in CQNCR. Figure 3 shows the average migration time by the heuristic, ABC, SA, and CQNCR for embedding VDC of the test cases. For CQNCR, we created an output configuration and CQNCR was run to discover a suitable pattern for migration. The average duration of migration using the proposed ABC and SA algorithms is 30% less compared to CQNCR and marginally less than the proposed heuristic algorithm. Simulated annealing shows slightly better results due to its capability to find a proper solution with the perturbation. Thus, the *MinCostMigration* solution reduces the possibility of SLA violations compared to the existing techniques. Heuristic algorithm falls behind because it either uses a greater number of costly migrations to house the new VDC or fails to find a feasible embedding.

Performance of a working application degrades while migrating the VM that hosts the application. Similar to the common SLA conditions on the availability of VMs, migrations standing longer than the permitted duration attracts penalty on the CDC. Figure 4 shows the comparison of the penalties for violating the SLA conditions for a sample VDC embeddings with live migrations wherein $ 0.2 per minute is charged for migration times exceeding 3 minutes. The savings on the penalty by employing the proposed ABC optimization is three times less than that of CQNCR. The heuristic algorithm is found suboptimal and hence results in a higher penalty in some cases. As stated earlier, the total payable penalty is related to the duration of the migration shown in Figure 3.

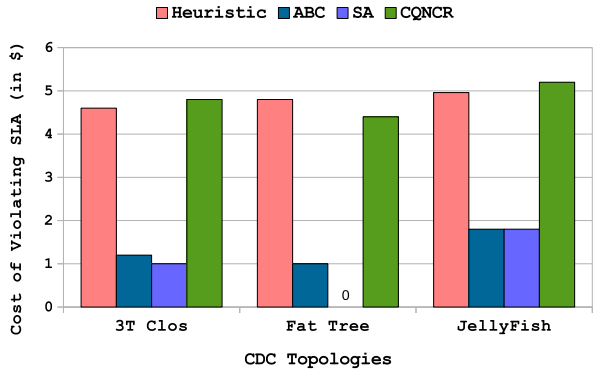


Figure 4: Comparison of the penal cost for migrations with test-cases for the heuristic, ABC, SA, and existing method.

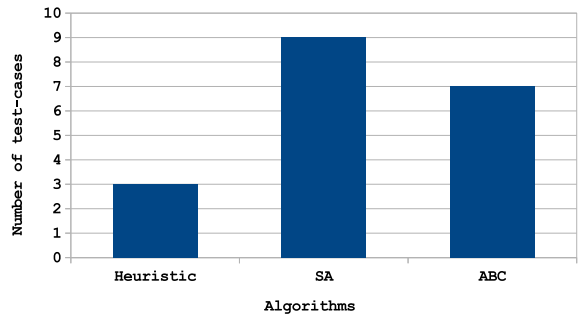


Figure 5: Comparison of the number of test-cases completed embedding within C_{max}.

Bounded cost C_{max} is the cost affordable by a CSP for SLA violation incurred due to migration while embedding a new VDC. For experimental evaluation, the bounded cost is set to $3 for the given test cases. Figure 5 shows the comparison of heuristics, ABC, and SA algorithms for the 10 test-cases. The graph shows the number of test cases completed within the bounded cost C_{max}. There is only one test case unmet by the simulated annealing algorithm and three test cases by ABC algorithm. These unsuccessful embeddings are considered to have exceeded the bounded cost C_{max}, and hence has a high cost.

Table 2 shows the execution time of Heuristic, SA, and ABC for the ten test-cases. Few test cases, for which the computation times are not available (indicated by '-'), are unsolved by the corresponding technique. Though the heuristic algorithm fails to find a bounded solution in most of the cases, the running time of the heuristic algorithm is much lesser than that of SA and ABC algorithms. However, considering the low acceptance rate and accuracy of results, meta-heuristics performs better than heuristics. It can be observed that the simulated annealing algorithm converges

Table 2: Execution Time (in seconds)

Test Case #	Heuristic	SA	ABC
1	0.7895	3.1209	7.6639
2	3.3114	6.378	10.9148
3	1.3647	7.6765	18.5543
4	0.5414	1.2544	9.0139
5	2.9745	4.8722	8.4759
6	0.7319	1.7132	7.713
7	0.7829	1.8507	11.9876
8	0.7045	1.3247	10.0745
9	-	7.1632	-
10	-	-	-

Table 3: Number of servers used for embedding a new virtual data center

Test Case #	Heuristic	SA	ABC
1	1	1	1
2	2	1	1
3	2	2	2
4	2	2	2
5	3	2	2
6	2	2	2
7	2	1	1
8	4	1	1
9	-	1	-
10	-	-	-

faster than ABC optimization. In practice, cloud management systems periodically consolidate servers and shutdown the vacated servers to reduce power consumption. Savings from the consolidation comes at a cost of violating SLAs during the process of live VM migration. Consolidating the residual resources at the time of embedding, it can reduce the overhead for migration for consolidation in the future. Proposed algorithms, both SA and ABC, consider consolidation of resources while embedding a new VDC, so that the CSP could save the cost of periodic consolidation. Moreover, consolidation whilst embedding helps to increase the rate of acceptance of VDCs because the consolidation adapts to the specific demands of the VDC being embedded. Table 3 shows the number of servers (physical machines) used for embedding the new VDC request. While the heuristic method uses more number of servers for embedding the new VDC, ABC and SA algorithm uses minimum number of servers, which reduces the requirement of consolidation in the future.

Finding the average of the total migration time may not be an invariably meaningful metric for comparing the techniques proposed above because, the lowest value of average would be associated with the heuristic technique, at the cost of optimality and feasibility. Figure 6 shows the sum of durations of migrations more than the tolerable limit as per the SLA. We illustrate this using a different dataset which has instances of VDCs which cannot be embedded. This dataset also has ten test cases each with a partially filled 3T Clos CDC and a new VDC. Data label NA (no allocation) means there is no feasible solution using the technique for the specific test case. A data label 0 means either a solution exists without migrations or the migrations do not violate SLA limits. CQNCR is not considered here because some of the test cases don't have a feasible solution, as required by CQNCR. The heuristic solutions have the highest values because the number of violating migrations in the solutions are higher compared to others. Simulated annealing algorithm is found to converge to the best migration sequence if there exists one.

We have showed that keeping a check on the number of costly migrations can reduce the penalty on the CSPs for violation of SLAs. SLA definitions can be different from the one used in this paper. For example, instead of the total migration time, in which the response of VM is slow, the actual downtime of the VMs can be taken to define the SLA. In any case, using VM migrations for optimal embedding

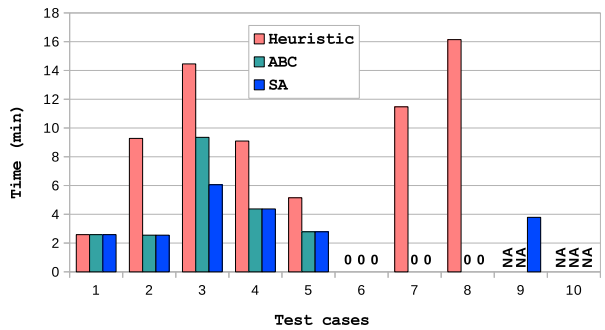


Figure 6: Sum of migration times exceeding the permitted duration while embedding a VDC.

of VDCs will be crucial when more and more users start using infrastructure-as-a-service(IaaS) cloud and the CSPs fall short of resources as per the user configurations. Apart from embedding new VDCs, the solutions proposed here can be employed for other resource demands, including scaling, with constraints.

6. CONCLUSION

This paper formally defines the problem of finding the most cost-effective set of VM migrations while serving a new VDC request on a cloud data center. Both the proposed meta-heuristic algorithms provide comparable optimal solutions with a difference in the computation time. However, we observed that the ABC optimization end up in suboptimal solutions and converges slowly. Experimental results show that minimizing the expensive VM migrations can be beneficial to avoid high penalty on SLA violations. The cost model for live migration is adapted from the existing literature with significant modifications to include all relevant parameters. The proposed model can be applied to any resource allocation problem in the CDC in addition to the VDC embedding problem. Identifying algorithms that find the migration pattern with additional constraints such as parallel migrations, in a reasonable amount of time to fit a real-time scenario, is a possible extension of this work. Scheduling the identified virtual machine migrations without incurring much time and cost is also a promising future work.

7. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, Aug. 2008.

[2] A. Amokrane, M. Zhani, R. Langar, R. Boutaba, and G. Pujolle. Greenhead: Virtual data center embedding across distributed infrastructures. *IEEE Trans. on Cloud Computing*, 1(1):36–49, Jan 2013.

[3] D. G. Anderson. Theoretical approaches to node assignment. *Unpublished Manuscript*, 2002.

[4] M. F. Bari, M. F. Zhani, Q. Zhang, R. Ahmed, and R. Boutaba. CQNCR: Optimal VM migration planning in cloud data centers. In *2014 IFIP Networking Conference*, pages 1–9, June 2014.

[5] J. Chase and D. Niyato. Joint optimization of resource provisioning in cloud computing. *IEEE Transactions on Services Computing*, 10(3):396–409, May 2017.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[7] M. P. Gilesh, S. D. M. Kumar, L. Jacob, and U. Bellur. Towards a complete virtual data center embedding algorithm using hybrid strategy. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2616–2617, June 2017.

[8] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 15:1–15:12, New York, NY, USA, 2010. ACM.

[9] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, October 2005.

[10] D. Karaboga and B. Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *Proceedings of the 12th International Fuzzy Systems Association World Congress on Foundations of Fuzzy Logic and Soft Computing*, IFSA '07, pages 789–798, Berlin, Heidelberg, 2007. Springer-Verlag.

[11] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5):975–986, Mar 1984.

[12] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[13] S. Nathan, U. Bellur, and P. Kulkarni. Towards a comprehensive performance model of virtual machine live migration. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, pages 288–301, New York, NY, USA, 2015. ACM.

[14] Nokia Siemens Networks GMBH & Co. KG, K. Hoffmann, and M. Hoffman. Associating computing resources and communication resources with a service in a resource management architecture, Aug 2012.

[15] L. Saino, C. Cocora, and G. Pavlou. A toolchain for simplifying network simulation setup. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SIMUTOOLS '13, Cannes, France, 2013. ICST.

[16] A. Strunk. Costs of virtual machine live migration: A survey. In *Proceedings of the 2012 IEEE Eighth World Congress on Services*, SERVICES '12, pages 323–329, Washington, DC, USA, 2012. IEEE Computer Society.

[17] D. Teodorovic, P. Lucic, G. Markovic, and M. D. Orco. Bee colony optimization: Principles and applications. In *2006 8th Seminar on Neural Network Applications in Electrical Engineering*, pages 151–156, Sept 2006.

[18] VMware. Service Level Agreement for VMware vCloud Air, 2015.

[19] F. Yan, T. T. Lee, and W. Hu. Congestion-aware embedding of heterogeneous bandwidth virtual data centers with hose model abstraction. *IEEE/ACM Transactions on Networking*, PP(99):1–14, 2016.

[20] Q. Zhang, M. Zhani, M. Jabri, and R. Boutaba. Venice: Reliable virtual data center embedding in clouds. In *2014 Proceedings IEEE INFOCOM*, pages 289–297, Toronto, April 2014.

[21] M. Zhani, Q. Zhang, G. Simon, and R. Boutaba. VDC Planner: Dynamic migration-aware Virtual Data Center embedding for clouds. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 18–25, Ghent, Belgium, May 2013.

ABOUT THE AUTHORS:

M. P. Gilesh is an Assistant Professor at Government Engineering College Wayanad. He obtained his Bachelor degree in Computer Science and Engineering from the University of Calicut and his M. E. degree in Computer Science and Engineering from Indian Institute of Science, Bangalore (2010). Currently he is pursuing his PhD at National Institute of Technology Calicut. His research interests include cloud computing, computer networks, and virtualization. He has presented several papers in top rated international conferences. He is a senior member of IEEE, member of ACM, and a life member of ISTE and CSI.

Sanjay Satheesh received his Bachelor Degree in Information Technology from Cochin University of Science and Technology, Kerala, India in 2015. He completed his M. Tech degree in Computer Science (Information Security) from National Institute of Technology Calicut, India, in 2018. Currently, he is working in Microfocus as an R&D Engineer. His research interest includes Virtulization and resource management in cloud.

S. D. Madhu Kumar is a Professor in the Department of Computer Science & Engineering of National Institute of Technology Calicut. He was the head of the department during 2012-2014. He received his PhD in Computer Science & Engineering from the Indian Institute of Technology Bombay (IIT Bombay). Earlier he received the degree of Master of Engineering in Computer Science \& Engineering from Indian Institute of Science, Bangalore, India. He has more than 50 International publications to his credit and has produced many PhDs. His research interests include Cloud Computing, Big Data Analytics, Software Engineering and DBMS. He is a senior member of IEEE and a member of ACM USA. He has been the chair and keynote speaker for many International Conferences. He has been the Track Co-chair of the Cloud Computing Track of the ACM Symposium on Applied Computing (SAC) since 2010.

Lillykutty Jacob obtained her B. Sc. (Engg.) degree from the Kerala University, in 1983; M. Tech. degree from the Indian Institute of Technology at Madras, in 1985; and PhD degree in Electrical Communication Engineering from the Indian Institute of Science, in 1993. She was with the department of Computer Science, Korea Advanced Institute of Science and Technology, S. Korea, for postdoctoral research; with the department of Computer Science, National University of Singapore, as visiting faculty; and also with the department of Computer Science and Engineering, University of South Florida, as an exchange visitor. Since 1985 she has been with the National Institute of Technology Calicut, where she is currently a Professor. She is a highly cited researcher with more than 150 publications in international journals and conferences. She serves as the Editorial Board member, reviewer, and TPC member for a number of IEEE and ACM journals and conferences. She is a Senior Member of IEEE.

Combining Deep Reinforcement Learning with Prior Knowledge and Reasoning

Nicolas Bougie
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda
Tokyo, Japan 101-8430
nicolas-bougie@nii.ac.jp

Li Kai Cheng
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda
Tokyo, Japan 101-8430
twnoken@gmail.com

Ryutaro Ichise
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda
Tokyo, Japan 101-8430
ichise@nii.ac.jp

ABSTRACT

Recent improvements in deep reinforcement learning have allowed to solve problems in many 2D domains such as Atari games. However, in complex 3D environments, numerous learning episodes are required which may be too time consuming or even impossible especially in real-world scenarios. We present a new architecture to combine external knowledge and deep reinforcement learning using only visual input. A key concept of our system is augmenting image input by adding environment feature information and combining two sources of decision. We evaluate the performances of our method in 3D partially-observable environments from the Microsoft Malmo platform. Experimental evaluation exhibits higher performance and faster learning compared to a single reinforcement learning model.

CCS Concepts

•**Computing methodologies** → **Reasoning about belief and knowledge; Sequential decision making;** *Neural networks;* Partially-observable Markov decision processes;

Keywords

Reinforcement Learning, Object Recognition, External Knowledge, Deep Learning, Knowledge Reasoning

1. INTRODUCTION

Reinforcement learning is a technique which automatically learns a strategy to solve a task by interacting with the environment and learning from its mistakes. By combining reinforcement learning and deep learning to extract features from the input, a wide variety of tasks such as Atari 2600 games [20] are efficiently solved. However, these techniques applied to 2D domains struggle in complex environments such as three-dimensional virtual worlds resulting a prohibitive training time and an inefficient learned policy.

A powerful recent idea to tackle the problem of computational expenses is to modularise the models into an ensemble of experts [17]. Since each expert focuses on learning a stage of the task, the reduction of the actions to consider leads to a shorter learning period. Although this approach is conceptually simple, it does not handle very complex environments and environments with a large set of actions.

A similar idea of extending the information extracted from low-level architectural modules [18] with high-level ones have been previously used in the area of cognitive systems [23] but does not directly relies on RL and was limited to a supervised classification problem. The idea was to leverage information about videos with external ontologies to detect events in videos.

A technique is called *Hierarchical Learning* [31][1] and is used to solve complex tasks, such as "simulating human brain" [16]. It is inspired by human learning which uses previous experiences to face new situations. Instead of learning directly the entire task, different sub-tasks are learned by the agent. By reusing knowledge acquired from the previous sub-tasks, the learning is faster and easier. Some limitations are the necessity to re-train the model which is time-consuming and problems related to catastrophic forgetting of knowledge on previous tasks.

Another approach, *Symbolic Reinforcement Learning* ([5], [3]) combines a system that learns an abstracted representation of the environment and high-order reasoning. However this has several limitations, it cannot support ongoing adaptation to new environments and cannot handle other sources of knowledge.

In this paper, our approach focuses on combining deep reinforcement learning and external knowledge. Using external knowledge is a way to supervise the learning and enhance information given to the agent by introducing human expertise. We augment the input of a reinforcement learning model whose input is raw pixels by adding high-level information created from simple knowledge about the task and recognized objects. We combine this model with a knowledge based decision algorithm using Q-learning [36] or a Support Vector Machine [7].

In our experiments, we demonstrate that our framework successfully learns in real time to solve a food gathering task and to find a target in a maze, in 3D partially observable environments by only using visual inputs. We evaluate our technique on two 3D environments built on top of the Malmo platform, Minecraft. Our model is especially suitable for tasks involving navigation, orientation or exploration, in

Copyright is held by the authors. This work is based on an earlier work: SAC'18 Proceedings of the 2018 ACM Symposium on Applied Computing, Copyright 2018 ACM, 978-1-4503-5191-1. http://dx.doi.org/10.1145/3167132.3167165

which we can easily provide external knowledge.

The paper is organized as follows. Section 2 gives an overview of reinforcement learning and most recent models. The environments are presented in Section 3. The main contribution of the paper is described in Sections 4. Results are presented in Section 5. Section 6 presents the main conclusions drawn from the work.

2. RELATED WORK

Below we give a brief introduction to reinforcement learning and the models used into our system architecture.

2.1 Reinforcement Learning

Reinforcement learning [11] consists of an agent learning a policy by interacting with an environment. At each time-step the agent receives an observation s_t and choose an action a_t. The agent gets a feedback from the environment called a reward r_t. Given this reward and the observation, the agent can update its policy to improve the future rewards.
Given a discount factor γ, the future discounted reward, called return R_t, is defined as follows :

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{1}$$

The goal of reinforcement learning is to learn to select the action with the maximum return R_t achievable for a given observation [30]. From Equation (1), we can define the action value Q^π at a time t as the expected reward for selecting an action a for a given state s_t and following a policy π.

$$Q^\pi(s,a) = \mathbb{E}\left[R_t \mid s_t = s, a\right] \tag{2}$$

The optimal policy is defined as selecting the action with the optimal Q-value, the highest expected return, followed by an optimal sequence of actions. This obeys the Bellman optimality equation:

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s^{'}, a^{'}) \mid s, a\right] \tag{3}$$

Q-learning [36] (Algorithm 1) is a straightforward technique to approximate $\pi \approx \pi^*$. The estimation of the action value function is iteratively performed by updating $Q(s,a)$. This algorithm is considered as an off-policy method since the update rule is defined as follow :

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma * max_a Q(s_{t+1},a) - Q(s_t,a_t)\right] \tag{4}$$

The choice of the action follows a policy derived from Q. The most common policy called ϵ-greedy policy trade-off the exploration/exploitation dilemma [32]. In case of exploration, a random action is sampled whereas exploitation selects the action with the highest estimated return. In order to converge to a stable policy, the probability of exploitation must increase over time.

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:
Sates $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0,1]$
Discounting factor $\gamma \in [0,1]$
procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  while Q is not converged **do**
    Start in state $s \in \mathcal{X}$
    while s is not terminal **do**
      Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x,a)$)
      $a \leftarrow \pi(s)$
      $r \leftarrow R(s,a)$
      $s' \leftarrow T(s,a)$
      $Q(s',a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s',a'))$
      $s \leftarrow s'$
  return Q

When the state space or the action space is too large to be represented, it is possible to use an approximator to estimate the action-value $Q^*(s,a)$:

$$Q^*(s,a) \approx Q(s,a;\theta) \tag{5}$$

Neural networks are a common way to approximate the action-value. The parameters of the neural network θ can be optimized to minimize a loss function L_i defined as the expected temporal difference error of Equation (3):

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[(y_i - Q(s,a;\theta_i))^2\right] \tag{6}$$

where $y_t = r_t + \gamma \max_{a'} Q_{\theta_{target}}(s_{t+1}, a^{'})$

The gradient of the loss function with respect to the weights is the following :

$$\nabla_{\theta i} L_i(\theta i) = \mathbb{E}_{s,a,r,s'}\left[(r + \gamma \max_{a'} Q(s^{'}, a^{'}; \theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta i} Q(s,a;\theta_i)\right] \tag{7}$$

Mnih *et al.* (2013) used this idea and created the famous method called Deep Q-learning (DQN) [20]. However, the learning may be slow due to the propagation of the reward to the previous states and actions.

Similarly, the value function $V^\pi(s)$ which represents the expected return for a state s following a policy π is defined as follows:

$$V^\pi(s) = \mathbb{E}\left[R_t \mid s_t = s\right] \tag{8}$$

Some reinforcement learning models such as *Actor-Critic* or

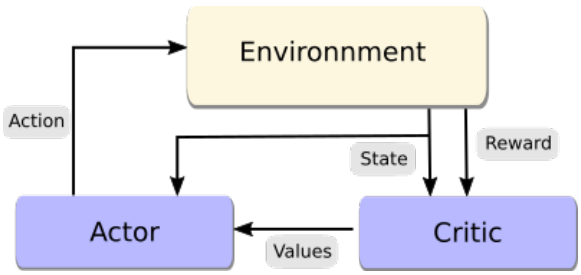


Figure 1: Actor-critic model

Dueling Network decompose the Q-values $Q(s,a)$ into two more fundamental values, the value function $V(s)$ and the advantage function $A(a,s)$ which is the benefit of taking an action compared to the others.

$$A(s,a) = Q(s,a) - V(s) \tag{9}$$

2.2 Asynchronous Advantage Actor-Critic (A3C)

It was shown that combining methods of deep learning and reinforcement learning is very unstable. To deal with this challenge, many solutions store the agent's data into a memory, then the data can be batched from the memory. It is done because sequences of data are highly correlated and can lead to learn from its mistakes resulting in a worse and worse policy. A3C [21] avoids computational and memory problems by using asynchronous learning. It allows the usage of on-policy reinforcement learning algorithms such as *Q-learning* [36] or *advantage actor-critic*. The learning is stabilized without using experience replay and the training time is reduced linearly in the number of learners.

The learners of A3C which use their own copy of the environment are trained in parallel. Each process will learn a different policy and hence will explore the environment in a different way leading to a much more efficient exploration of the environment than with a replay memory. A process updates its own policy based on an advantage actor-critic model [15] (Figure 1). The actor-critic model is composed by an actor which acts out a policy and a critic which evaluates the policy. The main thread is updated periodically using the accumulated gradients of the different processes.

The critic takes as input the state and the reward and outputs a score to criticize the current policy. In the case of advantage actor-critic model, the critic estimates the advantage function which requires to estimate V and Q.

The actor does not have access to the reward but only to the state and the advantage value outputted by the critic. Contrary to the critic which is value based, the actor directly works into the policy space and changes the policy towards the best direction estimated by the critic. Optimization techniques such as stochastic gradient descent are used to find θ that maximizes the policy objective function $J(\theta)$. The policy gradient objective function $\nabla_\theta J(\theta)$ is defined as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi,\theta}\left[\nabla_\theta log\pi_\theta(s,a)A^w(s,a)\right] \tag{10}$$

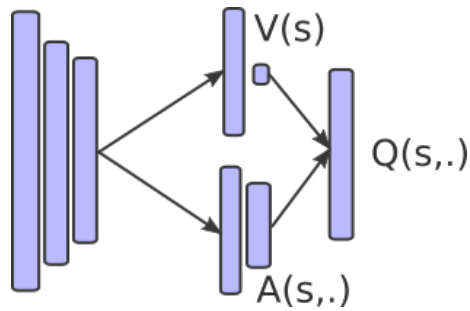


Figure 2: Dueling network architecture

where $A^w(s,a)$ is a long term estimation of the reward to allow the actor to go in the direction that the critic considers the best.

2.3 Dueling Network

The idea is to separately compute the advantage function and the value function and combine these two values at the final layer (Figure 2). The dueling network [35] may not need to care about both values and the advantage at any given time. The estimation of a state value is more robust by decoupling it from the necessity of being attached to a specific action. This is particularly useful in states where its actions do not affect the environment in any relevant way. For example, moving left or right only matters when a collision is upcoming. The second stream, which estimates the advantage function values, is relevant when the model needs to make a choice over the actions in a state. The Bellman's equation (3) becomes now:

$$Q(s,a;\theta,\alpha,\beta) = V(s,\theta,\beta) + (A(s,a;\theta,\alpha) - \max_{a'\in\mathcal{A}} A(s,a^{'};\theta,\alpha)) \tag{11}$$

And by changing the max by a mean:

$$Q(s,a;\theta,\alpha,\beta) = V(s,\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a^{'};\theta,\alpha)) \tag{12}$$

With θ the shared parameters of the neural network, α the parameters of the stream of the advantage function $\mathcal{A}$ and β the parameters of the stream of the value function V. Since the output of the two streams produces a Q function, it can be trained with many existing algorithms such as Double Deep Q-learning (DDQN) [34] or SARSA [26]. The main advantage is that for each update of the Q-values, the value function is updated whereas with traditional Q-learning only one action-value is updated.

2.4 Supervised Learning

In supervised learning, a set of input variables are used to predict other variables or classes. This machine learning task infers a function f mapping the input space X to the output space Y, $f : X \to Y$.

Unlike reinforcement learning, the function f is learned given

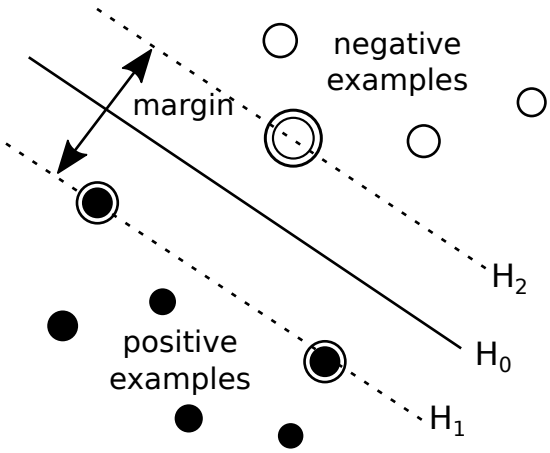


Figure 3: A linear Support Vector Machine

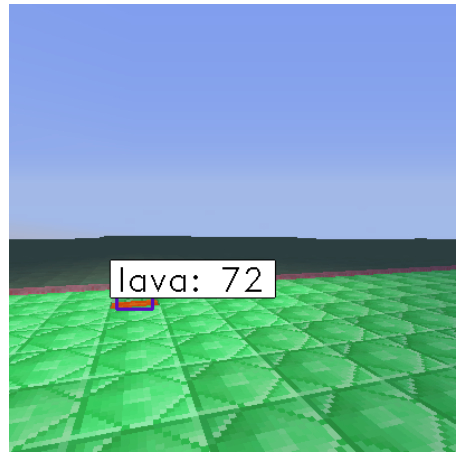


Figure 4: Screenshot of the environment

training examples $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$. In classification tasks, the output space is the set of the possible labels.

Support Vector Machine (SVM) [7][29] (Figure 3) is a supervised two-class classifier. Given an annotated dataset, it seeks a decision function $f(x)$ to determine the class of input x.

Linear SVM assumes that data are separable by a hyperplane $H_0 : w{\cdot}x{+}b = 0$ which separates the negative examples and positive examples with b a constant and w the normal vector to the hyperplane. We call margin the distance between the nearest positive example on $H_1 : w \cdot x + b = 1$ and the nearest negative example on $H_2 : w \cdot x + b = -1$. H_1 and H_2 are two hyperplanes parallel to H_0.

If the data are linearly separable, we obtain the following equation:

$$y_i(w \cdot x_i + b) - 1 \neq 0 \tag{13}$$

The SVM algorithm maximizes this margin by maximizing $\frac{1}{||w||}$. A dual quadratic programming problem can be obtained:

$$f(x) = w \cdot x + b = \sum_{i=1}^{l} \alpha_i y_i x_i \cdot x + b \tag{14}$$

with α Lagrange multipliers and $w = \sum_{i=1}^{l} \alpha_i y_i x_i$

In non separable linear data, the examples can be maped to an higher dimensional space using a kernel function [27], [9]:

$$f(x) = w \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \tag{15}$$

3. TASK & ENVIRONMENT

We built two environments on the top of the Malmo platform [10] to evaluate our idea. Malmo is an open-source platform that allows us to create scenarios with Minecraft engine. To test our model, we trained an agent to collect foods in a field with obstacles and to find a target in a maze. The agent can only receive partial information of the environment from his viewpoint. We only use image frames to solve the scenario. An example of screenshot with the object recognition results is shown in Figure 4.

3.1 Eating a Healthy Diet

The goal of the agent is to learn to have a healthy diet (Task 1). It involves to recognize the objects and learn to navigate into a 3D environment. The task consists in picking up food from the ground for 30 seconds. Food is randomly spread across the environment and four obstacles are randomly generated. Each of the 20 kinds of food has an associated reward when the agent picks it up. This reward is a number between +2 (healthy) and -2 (unhealthy). They are distributed equitably, meaning that a random agent should get a reward of 0.

The settings were: window size: *400* $\times$ *400* pixels, actions: *turn left, turn right, crouch, jump, move straight and move back* , number of objects: *200*, number of obstacles: *4*. The actions *turn left* and *turn right* are continuous actions to make the learning smoother as consecutive frames are more similar.

3.2 Finding a Target in a Maze

We also evaluate the algorithm on an orientation task: finding a target in a maze (Task 2). A positive reward (+1) is given when the agent reaches the goal and a negative reward is given in case of timeout (-900), after 45 seconds without finding the target. Several objects are randomly placed in the maze. An episode is restarted if the agent collides with a hazardous object and a negative reward is given (-1000). We generate the objects such as the target, represented by a flag, is always reachable.

The settings and the actions are the same as in the *Eating a healthy diet* task except for the number of different objects. We limit the possible objects to 15 including 5 hazardous objects. In total, 30 objects are generated at each episode.

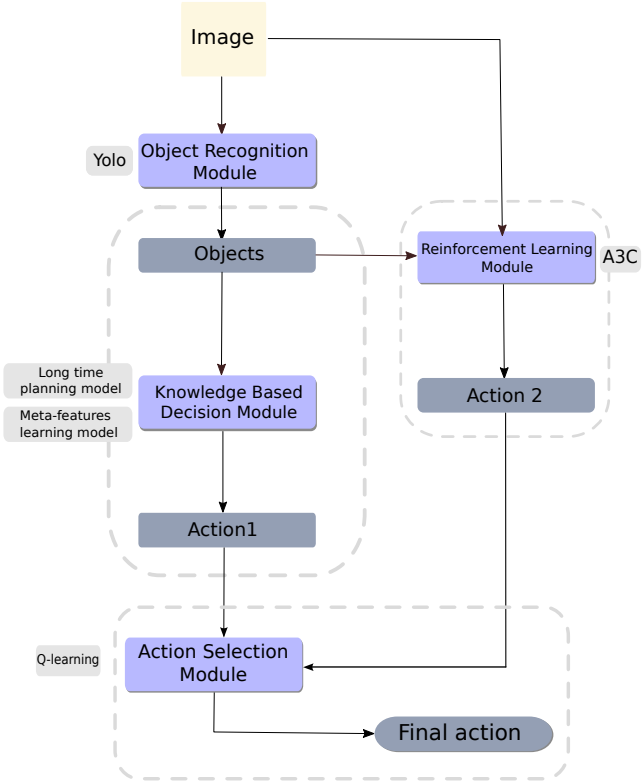


Figure 5: Global architecture of DRL-EK

4. SYSTEM ARCHITECTURE

4.1 General Idea

Figure 5 describes the global architecture of our new framework called DRL-EK. It consists of four modules: an Object Recognition Module, a Reinforcement Learning Module, a Knowledge Based Decision Module, and an Action Selection Module.

The object recognition module identifies the objects within the current image and generates high-level features. These features of the environment are then used to augment the raw image input to the reinforcement learning module. In parallel, the knowledge based decision module selects another action by combining external knowledge and the object recognition module outputs. To manage the trade-off between these two sources of decision we use an action selection module. The chosen action is then acted by the agent and the modules are updated from the obtained reward.

4.2 Object Recognition Module

Injecting external knowledge requires to understand the scene at a high-level in order to be interpreted by a human. The easiest way to understand an image is to identify the objects. For example, it is intuitive to give more importance to the actions *turn* or *jump* than the action *move straight* when an obstacle is in front of the agent. To recognize the objects, the module uses You Only Look Once (YOLO) [24][25] library which is based on a deep convolutional neural network. As input, we use an RGB image of size 400×400 pixels. YOLO predicts in real time the bounding boxes, the labels and confidence scores between 0 and 100 of the objects. An example is shown in Figure 4. We trained YOLO on a dataset of 25 000 images with twenty different classes corresponding to the food that is presented in the environment.

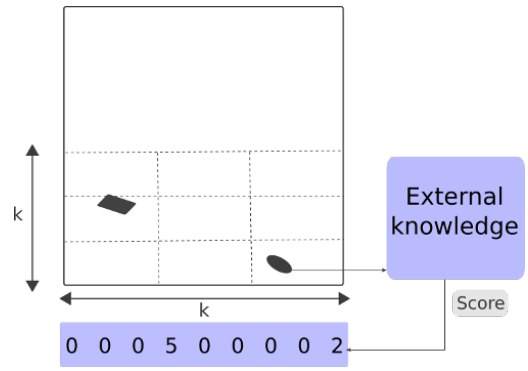


Figure 6: Important areas of an image

The model is trained off-line before starting the learning into the environment. The neural network architecture is adapted from the one proposed by Redmon *et al.* (2016) for the Pascal VOC dataset [25]. In order to recognize small objects, the size of cells is decreased from 7 to 5 pixels and the number of bounding boxes for each cell is increased from 2 to 4.

In addition to the identified objects, the module creates feature information about the current frame. To generate these high-level abstraction features we combine the recognized objects and external knowledge. They are then used as input by the reinforcement learning module and the knowledge based decision module. We designed three types of features *presence of objects*, *important area* and *important area with eligibility traces*.

4.2.1 *Presence Of Objects Features*

The first type of features is a vector of booleans which indicates whether an object appears or not within the current image. The size of this vector is the number of different objects in the environment. Since some objects are not helpful to solve the task, we can decide to only take some of the objects into account based on our knowledge about the task.

4.2.2 *Important Area Features*

As the position of objects is important, we encode information about objects within each area of the image. We split the image into k rectangles vertically and horizontally. So, the number of areas is k^2 and for each one we compute a score (Figure 6). The score of an area is the sum of the score of the objects within this area. External knowledge can be introduced by shaping the score of the objects. From our knowledge about the task, we manually defined the scores to indicate whether or not an object is important to solve the task. To tackle problems with partially observable environments, we keep track of recent information by concatenating the array of scores of the current frame with the arrays of

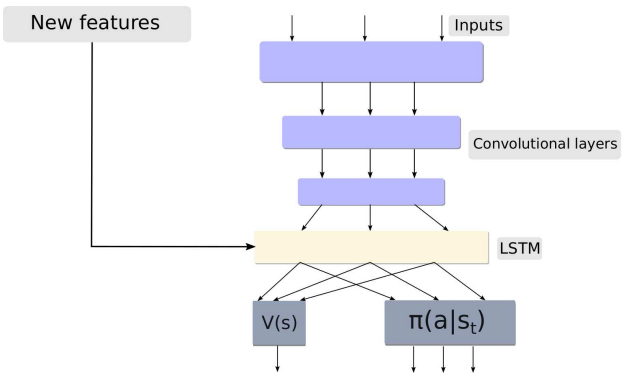


Figure 7: Injection of new features into the reinforcement learning module (A3C)

the two previous frames.

In our experiments, the top half of the images only contains the sky so we computed the important area features on the half bottom of the images. We gave a score of -15/+5 to foods we think is unhealthy (cake,cookie) / healthy (meat, fruit) and 0 for the others. That way, if an area contains a healthy food such as a fruit and a sweet food, then the score of the area will be lower than an area containing only a fruit or no object. We set the number of rectangles to 3 (9 areas in total: 3×3). We found that with a higher number of areas the amount of encoded information is bigger but information quality of each area is worse than with 3 areas.

4.2.3 Important Area Features with Eligibility Traces

Important areas with eligibility traces [28] features is a variant of the important area features (IAF) [2], suitable for orientation tasks such as *finding a target in a maze*. As with IAF, we compute a score for each part of the image but the score associated to each object is adapted over time depending on the observed rewards. The idea behind is that some objects are more important than the others to solve the task. Adapting the scores turned out to be critical in guiding the algorithm to solve tasks.

Our algorithm maintains a parametrized score-function which maps the objects to their score. At the beginning of the training, we initialize the scores using our prior knowledge. The scores are adapted over time to fit with the rewards. Instead of computing the value of each part of the image with the fixed scores, we use the updated scores of the objects.

Since objects have not an immediate impact on the reward, we use an eligibility trace mechanism to back-propagate the rewards. At each iteration, the new score $score(obj_i)'$ of an object obj_i is estimated as follows:

$$score(obj_i)' = score(obj_i) + \alpha(e_t(obj_i) \times r) \qquad (16)$$

with α the learning rate and $e_t(obj_i)$ the eligibility trace of obj_i:

$$e_t(obj_i) = \begin{cases} \lambda e_{t-1}(obj_i) + 1, & \text{if } s = s_t \text{ and } a = a_t \\ \lambda e_{t-1}(obj_i) & \text{otherwise} \end{cases} \qquad (17)$$

Our contribution here is to provide a simple technique to adapt our prior knowledge to fit the task.

We used a linear learning rate decay to learn the eligibility traces and an eligibility trace decay $\lambda = 0.5$.

4.3 Reinforcement Learning Module

For a computer, learning from an image is difficult and requires a lot of training steps. To deal with it, the entry point of most of the reinforcement learning models is a recurrent convolutional neural network [6] to extract temporal and spatial features of the image.

We trained a deep reinforcement learning model to perform policy learning and we modified the neural network structure to incorporate external knowledge. In addition to the image input, we injected *presence of objects* or *important area* features which are created by the object recognition module. In the neural network, we give to a Long Short Term Memory (LSTM) [8] the output of the last convolutional layer concatenated with the new features (Figure 7). The next layers of the neural network are two separated fully-connected layers to estimate the value function $V(s)$ and the policy $\pi(a|s_t)$. The purpose is to help the model at the beginning of the training to recognize and focus on objects. The new features augment the raw image input to the reinforcement learning model by adding high-level information. For example, from presence of objects features the model can decide which actions are allowed or not. If a *door* is detected some of the actions may become irrelevant such as *jumping*.

The choice of the reinforcement learning model highly depends on the environment. Since the model at each timestep takes an input and outputs an action, we can easily substitute most of the reinforcement learning techniques such as Deep Q-learning (DQN) [20], Deep Deterministic Gradient Policy (DDPG) [19], Dueling Network [35] or Asynchronous Actor-Critic Agents(A3C) [21] by using a recurrent convolutional neural network as state approximator.

A3C is the most suitable model to solve our task. We tested and empirically searched the best parameters such as a good convolutional neural network architecture and the choice of the optimizer of this model. It provides a baseline to evaluate the importance of each module of our architecture on the final policy.

Working directly with 400×400 pixel images is too computationally demanding. We apply image preprocessing before training A3C. The raw frame is resized to 200×200 pixels. To decrease the storage cost of the images we convert the image scale from $0 - 255$ to $0 - 1$.

We set the number of workers of A3C to 3 and a convolutional recurrent neural network is used to approximate the states. The reason why we use a recurrent neural network is because the environment is partially observable. The input of the neural network of A3C estimator consists in a 200×200×3 image. The 4 first layers convolve with the following parameters (filter: 32,32,32,32, kernel size: 8×8, 4×4,3×3,2×2, stride size: 2,2,2,1) and apply a rectifier nonlinearity. It is followed by a LSTM layer of size 128 to incorporate the temporal features of the environment. Two

separate fully connected layers predict the value function and a policy function, a distribution of probability over the actions. We use RMSProp [33] as optimization technique with $\epsilon = 10^{-6}$ and minibatches of size 32 for training.

4.4 Knowledge Based Decision Module

We believe that the agent is not able to accurately understand and take into account the objects of the environment. A human can easily understand and make a decision from high-level features such as the utility or name of an object. Getting this level of abstraction is difficult but we can help the machine by giving it less low-level information such as color of pixels but more high-level information such as the importance of an area of the image.

Moreover, when the reinforcement learning module is fed with the images and the presence of objects or important areas features, the training time is long due to the size of state space. The knowledge based decision module is able to select an action using external knowledge and high-level features generated by the object recognition module and without direct access to the image. We propose two different approaches, a knowledge reasoning model or a meta-feature learning model

4.4.1 Knowledge Reasoning Model

Our technique for solving the task relies on human reasoning and prior knowledge. Our previous approach [2] hard encodes a set of rules. This may require many modifications to be adapted to new environments. Our contribution here is to provide a new architecture able to store, retrieve and reason on knowledge regardless of the environment. In addition, abstracting rules from environments allow a better control and an easier transfer among domains.

To deal with environment abstraction, we divide the task into three steps: preprocessing, knowledge representation and reasoning.

The preprocessing file allows us to deal with complex and non-intuitive inputs. This step can be easily adapted to new environments. The purpose is to generate features that can be interpreted and stored. In case of images, a preprocessing can be the recognition of the objects within a frame. The output of the preprocessing file is then used to check which rules are satisfied. The rules are saved in a second file. They associate a pattern to an action and allow to introduce complex external knowledge about the task. An example of a simple rule is, if the object *cookie* is on the left of the image then the action *turn left* is forbidden.

A pattern is a conjunction of variables which can be arbitrarily complex. The variables represent significant events in the task. For example, in task involving driving a car, a variable could be *(speed between 20 and 50 km/h)* and an example of pattern is (*(speed between 20 and 50 km/h)* $\wedge$ *(pedestrian crossing the road)*).

Given an observation obs_t, the active rules are the rules for which all their variables are active. Given a priority order previously defined, one rule is selected.

The recommended action is selected in the reasoning file which combines traditional test-case algorithms and planning. It takes as input the selected rule and the output of the preprocessing file. We store in an array the sequence of planned actions. At each time-step, the model checks if the previously planned sequence of actions is still the optimal one and if it is not the case (for example the next action is *jump* but there is no obstacle) the algorithm updates it, otherwise the first action in the array is returned. If the selected action is related to the movement, the model estimates the best angle and the necessary number of steps to perform it. Finally, the first of the planned actions is returned.

To decrease the number of rules we discretized the image space into four areas: *center, left, right, other*. We designed 43 rules to prevent the agent from going in the direction of the food we think is dangerous. To avoid static behavior, we give more priority to the actions *turn left, turn right, move straight* than the others.

4.4.2 Meta-feature Learning Model

In our previous approach, we manually create rules to reason on high-level features. To automatically learn the rules and select the optimal action from them, we use a deep reinforcement learning model such as DQN or dueling network. Unlike the reinforcement learning module which uses the image, the only input is high-level features such as *important areas* or *presence of objects*. As the input is much smaller than an image, a simple neural network can be trained to approximate the states. The smaller number of parameters leads to a faster learning than a model trained from visual information.

In experiments, we trained a dueling network combined with a double deep Q-learning (DDQN). It empirically gives a smoother learning than most of the other reinforcement learning models. A neural network approximates the states. It consists in 3 fully connected layers of size 100 with a rectifier nonlinearity activation function. Network was trained using the *Adam* algorithm [14], learning rate of 10^{-3} and minibatches of size 32. As input, we used a slightly modified version of the important area features outputted by the object recognition module. To create important area features, we filtered the objects too far and the objects with a confidence score less than 0.25. Taking into account an object such as *grass* is irrelevant and makes the learning more difficult. We only used dangerous or very healthy (10 objects out of 20) objects and removed 2 objects that we know are difficult to distinguish.

4.5 Action Selection Module

The module aggregates the actions proposed by the reinforcement learning module and the knowledge based decision module to select the action that the agent will perform in the environment. The goal is to take advantage of the fast learning of the knowledge based decision module and the quality of the policy learned by the reinforcement learning module. An important aspect of the action selection, is selecting an action with the highest expected return but also detecting error patterns to correct them. An error must be detected when an action which has not been proposed could offer a higher *return*. In this section, we describe in detail a reinforcement learning approach and a supervised learning approach.

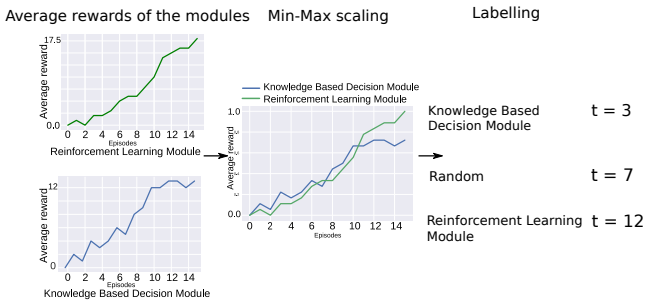


Figure 8: Data labeling using average reward as tag

4.5.1 *Reinforcement Learning for Action Selection*

We train a Deep Q-learning model to select the best action, detect and correct the error patterns. There is no restriction on the possible actions meaning that the final action may be different from the two proposed actions if an error is detected. We encode the proposed action by the two modules into two indicator vectors. An indicator vector is a binary vector with only one unit turned on to indicate the recommended action. The neural network input is the concatenation of these two vectors.

The Q-learning algorithm is trained using a Boltzmann distribution (Equation 18) as explorer and experience replay (each experience is stored into memory and the algorithm is run on randomly sampled batch) with a memory of size 10^6. Equation 18 gives the probability of selecting an action in a given state s.

$$P_s(a) = \frac{exp(Q(s,a)/\tau)}{\sum_{a' \in A} exp(Q(s,a')/\tau)} \tag{18}$$

The Q-network is composed of 2 hidden fully connected layers of size 50 and are followed by rectified linear units.

4.5.2 *Supervised Learning for Action Selection*

An alternative approach to the action selection problem is to train a supervised learning model. It restricts the problem to a classification task with two possible classes: *-1* reinforcement learning module and *+1* knowledge-based decision module.

We create the dataset during the training process. At each iteration, a new observation is added to the examples and we label them according to performance of each module (Figure 8). First, each module is evaluated to estimate its average reward over iterations. In order to compare the scores, we scale the rewards using a Min-Max normalization, with min, the minimum reward of the two average rewards and reciprocally for the maximum.

The class assignment is performed as follows:

$$P(class) = \begin{cases} -1 & \text{if } \beta_{RL} \geq \beta_{KBD} \\ 1 & \text{otherwise} \end{cases} \tag{19}$$

with $\beta_{RL_t} = \mathcal{N}(\bar{reward}(RL)_t, \sigma^2)$
and $\beta_{KBD_t} = \mathcal{N}(\bar{reward}(KBD)_t, \sigma^2)$ where $\sigma^2 = 0.05$ and $reward(x)_t$ the scaled average reward of the module x at the iteration t.

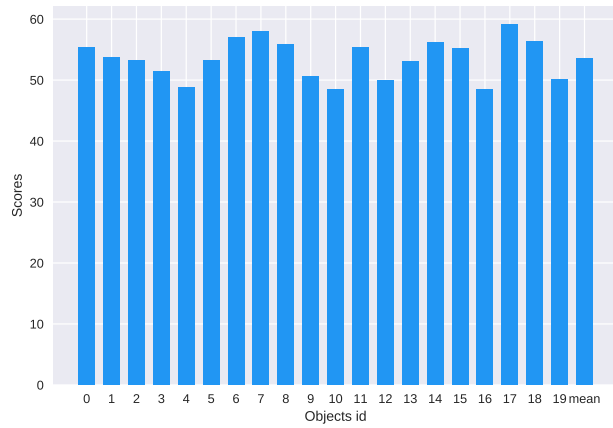


Figure 9: Average precision over all the classes obtained by the object recognition module (Task 1)

This means that we assign the label of the module with the highest average reward, and, when the rewards for both modules are similar we randomly assign a label.

To deal with the classification, we build a support vector machine trained with a stochastic gradient descent optimization. It allows us to update the model over time. The input variables are the same as with the *Reinforcement Learning* approach. We convert the predicted class by the SVM to an action and then act it in the environment.

The SVM was trained using a *l2* regularization term ([22]), a Gaussian kernel [12] and the penalty parameter $C = 5.0$

5. EXPERIMENTS

We conducted several experiments for evaluating our architecture. In all our experiments, we set the discount factor to 1.0. According to our different tests, on average the best reward that a perfect agent can get on task 1 in 30 seconds is 9.

5.1 Object Recognition

We evaluated our object recognition module for understanding the correctness of obtained object information in the environment.

Figure 10 reports the object recognition module performance on the *eating a healthy diet* task. In this experiment, we measured the mean average precision (mAP) as the error metric. The results are similar to the results presented by the authors (Redmon *et al.*, 2017) [25] on the Pascal VOC. dataset [4]. We obtained a mean average precision of 53.47. Although other libraries could offer higher performance, the real-time detection was the main criterion for selecting YOLO.

We noticed that most of the errors are false positives (68.3%) whereas the false negatives (31.7%) are uncommon. It leads to an agent with a policy more greedy and safer. As shown in the figure, the average precision is similar for every class.

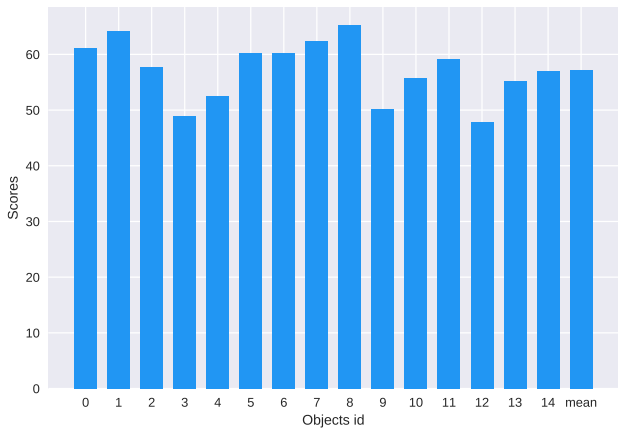


Figure 10: Average precision over all the classes obtained by the object recognition module (Task 2)

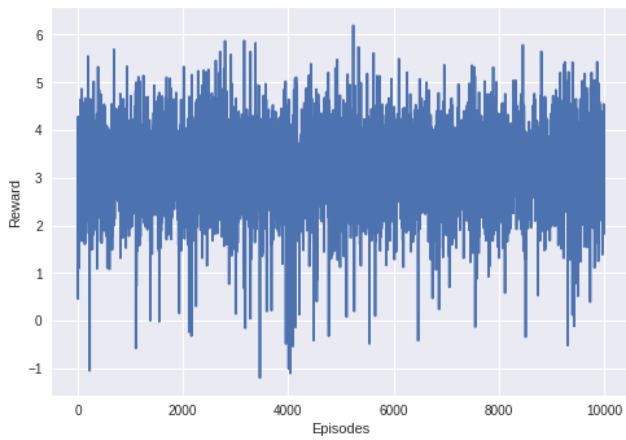


Figure 11: Evolution of the reward of the knowledge reasoning model

We also report the average precision for the objects of the task 2 (Figure 10). On this task, we obtained a mAP of 57.19. The best average precision was for the *object 8* (lava block) and the worse for the *object 12* (obsidian block) with an average precision of 47.9 and 65.2 respectively. The mean average precision is similar among the two tasks.

Note that the performance of YOLO is slightly affected by the complexity of the objects such as their shape, color or size. We could not establish a direct link between the visual complexity of an object and the average precision. We hypothesize that it relies on the quality of the training examples.

5.2 Knowledge Reasoning Model

Next, we tested the knowledge reasoning model in the knowledge based decision module to evaluate the effectiveness of this approach on the Task 1. We only utilized the object recognition module and the knowledge based decision module in our framework. The knowledge reasoning model performs much better than a random agent with an average reward of 3.3 (Figure 11), since expected reward of random agent is 0.

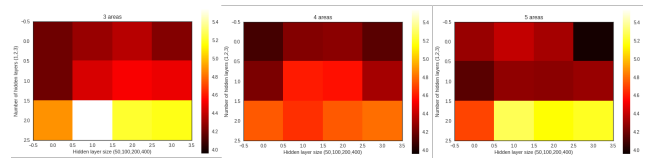


Figure 12: Average rewards of the meta-feature learning model with different parameters. The rewards were averaged over 200 episodes after 5000 training episodes

The most likely cause of the wide variance is the difficulty to handle all possible cases with manually created rules. For instance, the agent has difficulty in gathering food near obstacles. Since there is no learning, the quality of the agent only depends of the quality of the rules and is not able to converge. On the other hand, from the first episode the average reward is much higher than any other learning based models.

5.3 Meta-feature Learning Model

To evaluate performance of the meta-feature learning model we use as a baseline the knowledge reasoning model. We trained the model to gather food in a 3D environment (task 1).

We optimized its parameters, by sampling hyper-parameters from categorical distributions:

- Number of areas sampled from $\{4, 9, 16, 25\}$
- Number of hidden layers from $[1, 5]$
- Size of hidden layers sampled from $\{25, 50, 100, 200, 300\}$

Figure 12 reports an example of hyper-parameter optimization results. Each cell corresponds to a configuration of parameters. As can be seen on the figure, a number of hidden layers larger than two or a large number of areas results in lower performance. The best hyper-parameters are 9 areas, and a neural network with 3 hidden fully-connected layers of size 100. Training time is about 4 hours for each configuration on a Nvidia Titan-X GPU.

Figure 13 shows how the average total reward of the meta-feature learning model evolves during training with the optimal settings. The dueling network architecture effectively learns to solve the task from the important area features. The learning is fast during the first 3000 episodes and the average reward quickly converges around 5.4. It is also interesting to note that this approach rapidly achieves higher performance than the knowledge reasoning model. Automatic rule learning is more effective than manual rule construction. Unfortunately, the rules cannot be represented in a human-interpretable way.

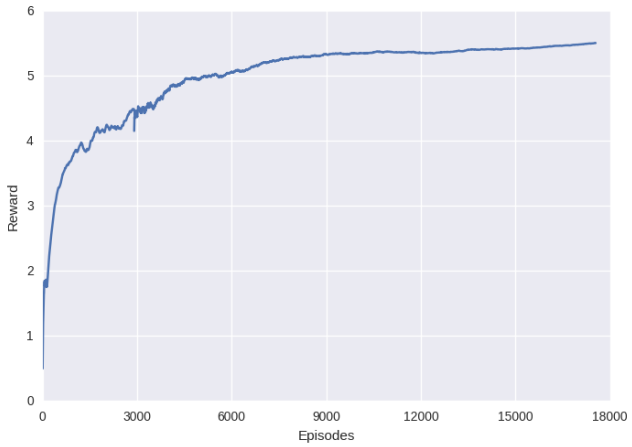


Figure 13: Average reward using meta-feature learning

Figure 14: Frequency of selection of each action

5.4 Action Selection

Table 1 reports the frequency of time the agent is able to reach the target in the maze (Task 2) using a supervised learning based action selection module and a reinforcement learning based action selection module. We obtained the results by running the corresponding agent 5 times. The agent was trained for 10000 episodes and then tested for 500 episodes.

Table 1: Frequency of time the agent reaches the target. The first row shows performance for a SVM as classifier and the second row for a Deep Q-learning algorithm. The last row consists in a random action selection between the two proposed actions.

Settings	Frequency
SVM	0.421
Deep Q-learning	0.624
Random	0.372

The Deep Q-learning method outperforms the SVM technique and a random selection. In particular, low performance of the supervised approach can be caused by the difficulty to annotate the dataset. However, further analysis shows that the action selection module, trained with a SVM exhibits higher performance during first 4000 iterations than with a DQN. This indicates that the lack of adaption capability of the SVM is harming the quality of the resulting policies.

We also evaluated the characteristics of the action selection module (task 1). We report the percentage of actions which is selected from the knowledge based decision module (action 1) and from the reinforcement learning module (action 2) against other actions. We measured the frequency of selection of each action every 350 episodes. As shown in Figure 14 the action selection module at the beginning selects equally the actions then more the action 1 and gradually give more importance to the action 2. The results confirm our intuition, the module selects the action of the most efficient module and adapts over time the trade-off between the sources of decision to always select the best one.

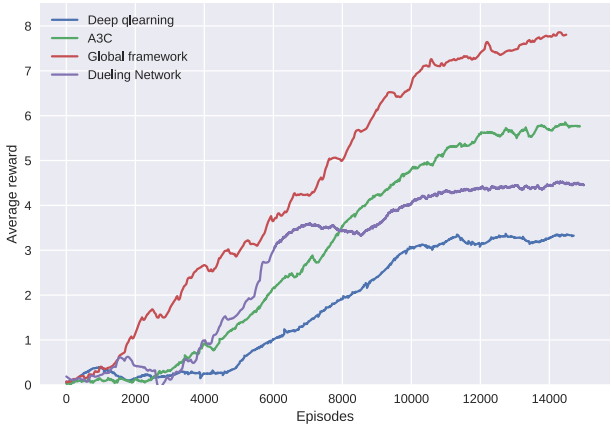


Figure 15: Performance of DRL-EK comparing to DQN, Dueling Network and A3C

5.5 Global Model Evaluation

Finally, we report the average reward of our whole framework trained using the injection of important areas features with eligibility traces into the reinforcement learning module and a meta-feature learning model as knowledge based decision module. A Deep Q-learning was used to select the best action. Figure 15 compares our proposed method with the best performing reinforcement learning methods on the Task 1. These models learn the policy only using raw pixels.

DRL-EK boosts A3C by injecting important area features with eligibility traces. To select these features, we compared *A3C+presence of objects features*, *A3C+important area features*(A3C*) and *A3C+important area features with eligibility traces* (Table 2). In both cases, the results show that adding a new input to the reinforcement learning module improves the quality of the policy.

Table 2: The table compares average reward for various features injected into A3C. The reinforcement learning module was evaluated alone for 12 000 episodes.

Settings	Rewards
A3C	5.6
A3C + *presence of objects features*	5.8
A3C + *important area features* (A3C*)	6.1
A3C* + *eligibility traces*	7.3

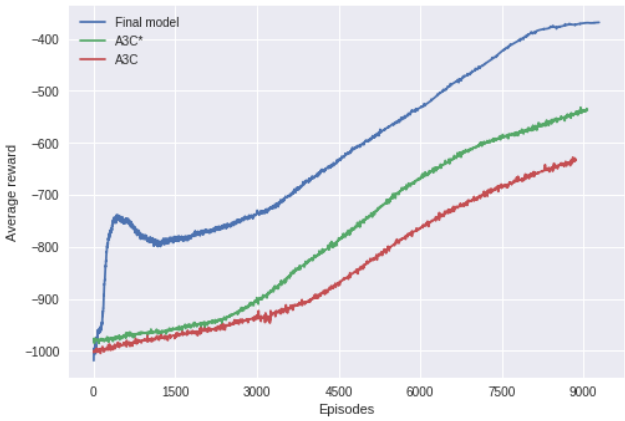


Figure 16: Comparison of performance curves between DEL-EK, A3C, A3C*.

As can be seen, DQN gives the worst results with an average reward of 3.2, $\approx 40\%$ less than A3C after converging. After 12 000 episodes, the average reward of the dueling network architecture trained with a double deep Q-learning is around 4.4 while A3C is able to achieve an average reward of 5.6. Surprisingly the meta-feature learning model trained alone (Figure 13) achieves higher performance than learning only from the image with a dueling network or a DQN model.

Asynchronous advantage actor-critic tends to learn faster than any other reinforcement learning based models. We believe this is due to the 3 parallel workers of A3C which offer a nonlinear significant speedup.

These results show that our architecture, DRL-EK, outperforms the baselines. Its average reward is around 15% better than A3C after 14 000 episodes. Moreover, the performance at the beginning of the training and the learned policy of DRL-EK is significantly better than all other models. One thing to note is that the action selection module tends to select an action different from action 1 and 2 (Figure 14). The continuous increase of the average reward of DRL-EK and this observation indicates that the action selection module is partially able to learn to correct the errors.

We also report learning curve on the Task 2 (Figure 16). We observe similar results as in the task 1. DRL-EK converges to an higher rewards and clearly improves over A3C and A3C*. In average, our model is able to reach the target is about 60% of time whereas A3C and A3C* achieve a score of 40% and 50% respectively.

The experiments demonstrate the importance of each module of our system. With an average time for one step of 0.43 seconds on a Nvidia Titan-X (Pascal) GPU, DRL-EK can be trained in real time.

6. CONCLUSION

We proposed a new architecture to combine deep reinforcement learning with external knowledge. We demonstrated its ability to solve complex tasks in 3D partially observable environments with image as input. Our central thesis is enhancing the image by generating high-level features of the environment. Further benefits stem from efficiently combining two sources of decision and adapting prior knowledge to fit the tasks. Moreover, our approach can be easily adapted to solve new tasks with a very limited amount of human work. We have demonstrated the efficacy of our architecture to decrease the training time and to learn a better and more efficient policy.

In the future, a promising research area is building an agent that creates a symbolic representation of the environment. Another challenge is how to integrate complex and structured external knowledge such as ontologies or textual data into our model. Finally, we are interested in extending our experiments to new environments such as VizDoom [13].

7. REFERENCES

[1] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[2] N. Bougie and R. Ichise. Deep Reinforcement Learning Boosted by External Knowledge. *ArXiv e-prints*, December 2017.

[3] A. d'Avila Garcez, A. Resende Riquetti Dutra, and E. Alonso. Towards Symbolic Reinforcement Learning with Common Sense. *ArXiv e-prints*, April 2018.

[4] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[5] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards Deep Symbolic Reinforcement Learning. *arXiv preprint arXiv:1609.05518*, 2016.

[6] Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015.

[7] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[9] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[10] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, NY, USA, 9-15 July 2016*, pages 4246–4247. IJCAI/AAAI Press, 2016.

[11] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[12] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[13] Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Proceedings of IEEE conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[15] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Proceedings of Neural Information Processing Systems*, pages 1008–1014. The MIT Press, 1999.

[16] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, pages 1–101, 2016.

[17] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Proceedings of AAAI*, pages 2140–2146, 2017.

[18] Antonio Lieto, Christian Lebiere, and Alessandro Oltramari. The knowledge level in cognitive architectures: Current limitations and possible developments. *Cognitive Systems Research*, 2017.

[19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ArXiv e-prints*, September 2015.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *ArXiv e-prints*, December 2013.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pages 1928–1937, 2016.

[22] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[23] Oltramari and Lebiere. Using ontologies in a cognitive-grounded system: Automatic action recognition in video-surveillance. 2012.

[24] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. volume abs/1506.02640, 2015.

[25] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[26] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, University of Cambridge, October 04 1994.

[27] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[28] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

[29] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[31] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 1553–1561, 2017.

[32] Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

[33] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, April 2012.

[34] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[35] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1995–2003, 2016.

[36] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

ABOUT THE AUTHORS:

Nicolas Bougie graduated from the University of Paris Sud, France in 2017. He studied about machine learning and artificial intelligence. He is currently a PhD student at the National Institute of Informatics in Japan and a student at the Sokendai University. His research area covers reinforcement learning, deep learning and machine learning.

Li Kai Cheng graduated from National Tsing Hua University, Taiwan in 2017. He studied about systems and their applications. His research topic at the National Institute of Informatics is machine learning and big data. His research interests include artificial intelligence, machine learning and data mining.

Ryutaro Ichise received his Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2000. From 2001 to 2002, he was a visiting scholar at Stanford University. He is currently an associate professor in Principles of Informatics Research Division at National Institute of Informatics in Japan. His research interests include machine learning, semantic web, and data mining.

An Extended Flexible Job Shop Scheduling Problem with Parallel Operations

Willian Tessaro Lunardi
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
29 Avenue J.F Kennedy L-1855
Luxembourg City, Luxembourg
willian.tessarolunardi@uni.lu

Holger Voos
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
29 Avenue J.F Kennedy L-1855
Luxembourg City, Luxembourg
holger.voos@uni.lu

ABSTRACT

Traditional planning and scheduling techniques still hold important roles in modern smart scheduling systems. Realistic features present in modern manufacturing systems need to be incorporated into these techniques. Flexible job-shop scheduling problem (FJSP) is one of the most challenging combinatorial optimization problems. FJSP is an extension of the classical job shop scheduling problem where an operation can be processed by several different machines. In this paper, we consider the FJSP with parallel operations (EFJSP) and we propose and compare a discrete firefly algorithm (FA) and a genetic algorithm (GA) for the problem. Several FJSP and EFJSP instances were used to evaluate the performance of the proposed algorithms. Comparisons among our methods and state-of-the-art algorithms are also provided. The experimental results demonstrate that the FA and GA achieved improvements in terms of efficiency and efficacy. Solutions obtained by both algorithms are comparable to those obtained by algorithms with local search. In addition, based on our initial experiments, results show that the proposed discrete firefly algorithm is feasible, more effective and efficient than our proposed genetic algorithm for the considered problem.

CCS Concepts

•Theory of computation → Discrete optimization; Evolutionary algorithms; Optimization with randomized search heuristics;

Keywords

Firefly algorithm, Genetic algorithm, Swarm Optimization, Evolutionary Algorithm, Flexible job-shop scheduling problem

1. INTRODUCTION

Production scheduling is one of the most important issues in the planning and scheduling of modern manufacturing systems. There are several workshop styles in the manufacturing system (including the Job Shop Scheduling Problem, JSP). The JSP can be stated as follows. Consider a set of machines and a set of jobs. Each job consists of a sequence of operations to be processed in a given order. Each operation must be processed individually on a specific machine. The objective is to find a processing sequence for each machine that minimizes an objective function, e.g. the completion time of the last operation (makespan).

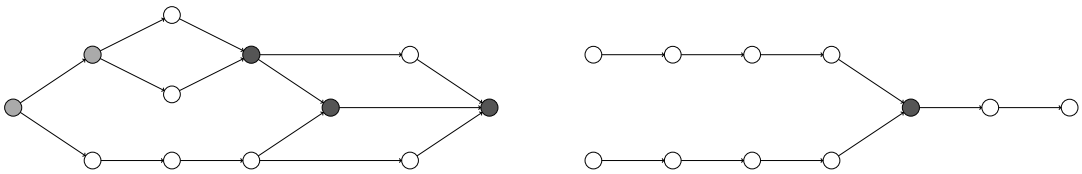
The inexistence of flexibility on the resources of each operation presented on JSP may meet the requirements of a traditional manufacturing system. However, with the ushering of the fourth industrial revolution (Industry 4.0) many computing devices, flexible manufacturing systems, and numerical control machines are introduced in order to achieve a higher level of autonomously, customization, and flexibility. Therefore, the assumption that one machine only processes one type of operation works in JSP but does not reflect the reality of the modern manufacturing systems.

The Flexible Job Shop Problem (FJSP) is an extension of the classical JSP problem where is considered that there may be several machines, not necessarily identical, capable of processing an operation. Particularly, for each operation, a set of machines on which that operation can be processed is given. The goal is to decide on which machine each operation will be processed and in what order the operations will be processed on each machine so that the makespan is minimized.

In terms of computational complexity, JSP problem is known to be one of the most difficult combinatorial optimization problems [10], and has been proven to be an NP-hard problem [8]. Since the FJSP and EFJSP problems are at least as difficult as the JSP, both are also NP-hard. Many methods have been presented to solve the FJSP problem, e.g. exact methods [6, 5, 1], and heuristic methods [15, 17, 7, 11, 13, 14].

In the literature, each job in the FJSP consists of a simple sequence of operations, so-called *path-jobs*. In some industrial environments, it is common to have jobs whose operations can be processed simultaneously. Mutually independent sequences of operations may feed into an "assembling" operations. Similarly, there may be "disassembling" operations which split the sequences of subsequent operations into two or more mutually independent sequences, so-called *G-job*.

Copyright is held by the authors. This work is based on an earlier work: SAC'18 Proceedings of the 2018 ACM Symposium on Applied Computing, Copyright 2018 ACM 978-1-4503-5191-1. http://dx.doi.org/10.1145/3167132.3167160



(a) G-job type. (b) Y-job type.

Figure 1: Types of jobs present in the FJSP with parallel operations (EFJSP).

Figure 1a shows a representation of G-job. Moreover, some jobs may consist of two independent sequences of operations followed by a third that puts together the results of the first two, so-called *Y-job*. Figure 1b shows a representation of Y-job. This problem will be referred to as "EFJSP" in this paper. Figure 1 shows the types of jobs present in the EFJSP, where each node represents an operation. The arcs represent precedence constraints and all arcs are directed from left to right. The black nodes are assembling operations and gray nodes are disassembling operations.

In recent years, several evolutionary algorithms (EA) were proposed to solve the FJSP. Recently, in [3], it was shown that hybrid techniques have been applied more often than other methods for solving FJSP. Furthermore, the technique most frequently chosen for performing exploration in hybrid algorithms is the genetic algorithm (GA). Although GA has powerful global searching ability [11], a recently developed algorithm called the Firefly Algorithm (FA), proposed in [16], has been shown [12, 16] to outperform Particle Swarm Optimization (PSO) and GA for continuous optimization problems.

In this paper, we propose and compare a Genetic Algorithm and a discrete Firefly Algorithm for solving the EFJSP. Several FJSP and EFJSP instances are used to evaluate the performance of both methods. Through experimental studies, the merits of each algorithm are demonstrated clearly. Furthermore, the proposed algorithms are compared with other state-of-the-art algorithms. The remainder of this paper is structured as follows. The problem formulation is discussed in Section 2. The solution representation for both algorithms is defined in the Section 3. The discrete FA is proposed in Section 4 and the GA is proposed in Section 5. Experimental results related to the proposed approaches are reported in Section 6. Section 7 addresses the conclusions and potential future works.

2. PROBLEM FORMULATION

Let (V, A) be a directed acyclic graph (DAG), where the vertices represent the operations, and the arcs represent precedence constraints. We are also given a set M of machines and a function F that associates a non-empty subset $F(v)$ of M with each operation v. The machines in $F(v)$ are the ones that can process operation v. Additionally, for each operation v and each machine k in $F(v)$, we are given a positive rational number p_{vk} representing the processing time of operation v on machine k. A machine assignment is a function f that assigns a machine $f(v) \in F(v)$ with each operation v. Given a machine assignment f, let $p_v^f := p_{v,f(v)}$.

For each machine k, let V_k be the set of operations that can be processed on machine k, that is, $V_k = \{v \in V : k \in F(v)\}$. Let B_k be the set of all ordered pairs of distinct elements of V_k. The pairs (v, w) in B_k are designed to prevent v and w from using machine k at the same time. Let B denote the union of all B_k. Hence, $(v, w) \in B$ if and only if $v \neq w$ and $F(v) \cap F(w) \neq \emptyset$.

Given a machine assignment f, let B^f be the set of all ordered pairs of distinct operations to be processed on the same machine, that is, $B^f = \{(v, w) \in B : f(v) = f(w)\}$. A *selection* is any subset Y of B^f such that, for each $(v, w) \in B^f$, exactly one of (v, w) and (w, v) is in Y. A selection corresponds to an ordering of the operations to be processed on the same machine. A selection Y is admissible if $(V, A \cup Y)$ is a dag.

Given a machine assignment f and a admissible selection Y, a *schedule* for $(V, A \cup Y, p^f)$ is a function s from V to the set of non-negative rational number such that $s_v + p_v^f \leq s_w$ for each (v, w) in $A \cup Y$. The number s_v is the starting time of operation v. The length of a (directed) path $(v_1, v_2, ..., v_l)$ in the dag $(V, A \cup Y)$ is the number $p_{v_1}^f + p_{v_2}^f + ... + p_{v_l}^f$. For any path P in $(V, A \cup Y)$ ending at v and any schedule s, the length of P is at most s_v. For each v in V, let s_v^* be the maximum of the lengths of all paths in $(V, A \cup Y)$ ending at v. There is a simple dynamic programming algorithm that computes the tight schedule [4]. Not surprisingly, the makespan of the tight schedule s^* is determined by the longest path: there exists a path $P = (v_1, v_2, ..., v_l, v_{l+1})$ in $(V, A \cup Y)$ such that the length of P plus $p_{v_{l+1}}$ equals $mks(s^*)$ (such P is known as a *critical path*).

Table 1: EFJSP instance with three jobs and a total of ten operations and four machines. Job 1 is a path-job, job 2 is a Y-job, and job 3 is a G-job. The column DFS represents the topological order of each respective job given by depth-first search algorithm.

Job	v	$p_{v,1}$	$p_{v,2}$	$p_{v,3}$	$p_{v,4}$	Route	DFS
1	1	2	3	4	3	$1 \rightsquigarrow 2$	
	2	3	5	2	2	$2 \rightsquigarrow 3$	$1, 2, 3$
	3	5	1	4	4	-	
2	4	4	3	4	5	$4 \rightsquigarrow 6$	
	5	3	3	4	2	$5 \rightsquigarrow 6$	$4, 5, 6$
	6	4	3	1	4	-	
3	7	3	1	3	3	$7 \rightsquigarrow 8, 9$	
	8	5	3	1	3	$8 \rightsquigarrow 10$	$7, 9, 8, 10$
	9	4	4	2	5	-	
	10	3	5	4	4	-	

$v \rightsquigarrow u$ represents that v precedes u in the job route.

V	1	2	3	4	5	6	7	8	9	10
MS	1	4	2	1	4	3	2	3	3	1
OS	3	1	2	3	2	3	1	3	2	1
TOS	7	1	4	9	5	8	2	10	6	3

Figure 2: Representation of a EFJSP solution.

2.1 EFJSP MILP Model

The MILP model for the EFJSP proposed in [1] can be given as follows: find a rational number z, rational arrays s and p', and binary arrays x and y that

$$\begin{aligned}
&\text{Minimize } z \\
&\text{subject to} \\
&s_v + p'_v \leq z && \forall v \in V, && (1)\\
&\sum\nolimits_{k \in F(v)} x_{vk} = 1 && \forall v \in V, && (2)\\
&p'_v = \sum\nolimits_{k \in F(v)} p_{vk} x_{vk} && \forall v \in V, && (3)\\
&y_{vw} + y_{wv} \geq x_{vk} + x_{wk} - 1 && \forall k \in M, \forall (v,w) \in B_k, && (4)\\
&s_v + p'_v \leq s_w && \forall (v,w) \in A, && (5)\\
&s_v + p'_v - (1 - y_{vw})L \leq s_w && \forall (v,w) \in B, && (6)\\
&s_v \geq 0 && \forall v \in V. && (7)
\end{aligned}$$

As x is binary, constraint (2) ensures that x is a machine assignment. Then constraint (3) makes array p' represent the processing times of operations. In fact, p' can be seen as an intermediate value, not a variable, that helps to simplify the presentation of the model. Since $p_{v,k} > 0$ for all v and k, thus $p'_v > 0$ and so constraint (6) makes sure that y_{vw} and y_{wv} are not both equal to 1. Hence, as y is binary, constraint (4) implies that y represents a selection. Indeed, if $x_{vk} = x_{wk} = 1$, which means v and w are assigned to machine k, then (4) forces y to decide whether v comes before or after w. Otherwise, constraint (4) is trivially satisfied. Once y is a selection and p' represents the processing times, constraints (5), (6), and (7) make s represent a schedule. Finally, the objective function and constraint (1) make sure z is the makespan of the schedule, and is as small as possible. Finally, L is an upper bound on the makespan of an optimal solution of the FJSP problem.

3. ENCODING AND DECODING

Since the FJSP is composed of two sub-problems, its solution representation is composed of two strings, i.e., machine assignment string (MS) and operation sequence string (OS). The MS string denotes the assigned machine for each particular operation and the OS string represents the order in which the operations are to be processed in their assigned machines. We denote ζ_{ime} as the eth element of the mth string of the ith individual (or firefly), where ζ_{i1} and ζ_{i2} are respectively the MS string and the OS string. Figure 2 shows a solution for the given set of machines and operations presented in Table 1. The solution ζ_i is composed of the MS string (ζ_{i1}) and the OS string (ζ_{i2}). The OS is translated onto TOS string and it maps the appearances of job id and operation.

Algorithm 1 Longest path $l(0, *)$

```
T ← topological sort of V̲
for each v ∈ T do
    E ← out-edges of v
    for each ε ∈ E do
        u ← target vertex of ε
        if s_u < s_v + p_{v,f(v)} then
            s_u ← s_v + p_{v,f(v)}
        end if
    end for u
end for v
```

3.1 Machine Assignment

Each element of the MS string is associated with a unique operation and the eth element represents the selected machine k for the corresponding operation V_e, where the index V_e denotes the eth operation in V and $k \in F(V_e)$. For example, on Figure 2, ζ_{i13} expresses that the machine 2 is assigned for operation 3. The length of the MS string is equal to $\sum_{i=1}^{n} J_i$, where J_i is the number of operations of job i and n is the number of jobs. The index does not vary throughout the whole searching process. Figure 2 shows an example of MS strings based on the given set of machines and operations presented in Table 1.

3.2 Operation Sequence

The OS string represents the order in which the operations will be processed in their designated machines. To avoid repair mechanisms, this representation uses an unpartitioned permutation with J_i repetitions of the job numbers, i.e., the index of job i appear in the string J_i times. Figure 2 displays an example of OS strings based on the jobs shown in Table 1.

3.3 Translating the OS String

By scanning the OS string from left to right, the f_ith appearance of a job i refers to the f_ith operation in topological order of the operations of job i. For example, scanning the OS string from left to right, for every appearance of job i, f_i is increased by 1, and the f_ith operation on the topological order is added to the TOS string. The translation mechanism bypasses the use of repair mechanism since any permutation can be decoded into a DAG, leading to a feasible solution. For the instance shown in Table 1, one possible OS string, and its TOS string is presented in Figure 2.

3.4 Calculation of the Makespan

Following the translation of the OS string and the assignment based on the MS strings, the starting time of every operation has to be defined in order to achieve the makespan (mks) of the solution. The calculation of the mks can be done by using graph traversal algorithms, commonly used in temporal planning. Additionally to the notations given in Section 2, two dummy nodes are introduced. In this way, the set of vertices is $\underline{V} = V \cup \{0, *\}$, 0 being the source and *

Table 2: Movement update of ζ_i towards a brighter firefly ζ_j.

	MS string	OS string
Firefly ζ_j	1 4 2 1 4 3 2 3 3 1	3 1 2 3 2 3 1 3 2 1
Firefly ζ_i	2 4 3 1 2 3 2 3 1 1	3 2 1 3 2 3 1 2 1 3
H_{ij} and S_{ij}	{1, 3, 5, 9}	{(2,3), (8,10), (9,10)}
$\underline{d}_{ij}$ and $\overline{d}_{ij}$	$\lvert 2-1\rvert + \lvert 3-2\rvert + \lvert 2-4\rvert + \lvert 1-3\rvert = 6$	$\lvert 2-3\rvert + \lvert 8-10\rvert + \lvert 9-10\rvert = 4$
Attractiveness $\beta(r)$	0.217	0.384
$\underline{R}$ and $\overline{R}$	{0.13, 0.78, 0.17, 0.24}	{0.35, 0.72, 0.11}
Position after β-step	**1** 4 3 1 **3** 3 2 3 1 1	3 **1** **2** 3 2 3 1 2 **3** **1**
Position after α-step	1 4 3 1 3 3 2 3 1 **4**	3 1 2 **2** **3** 3 1 2 3 1

being the sink node, respectively representing the start and the end of the planning period.

The length of a path is defined as $\sum_{i=1}^{q-1} \mu(v_i, v_{i+1})^f$, where q is the number of nodes in the path, v_i is the ith node, and $\mu(v, u)$ gives maximal length of all edges between node v and u. Denoting the value of some longest path from node v to node u by $l(v, u)$, the starting time of operation v_i in a left justified schedule is equal to $l(0, v_i)$ in the corresponding solution graph. Therefore, the mks of a solution is equal to the length of some longest path from 0 to *, $l(0, *)$. The pseudo code shown in Algorithm 1 summarizes the steps to calculate the longest path $l(0, *)$.

4. FIREFLY ALGORITHM

Firefly algorithm (FA) is a simple yet quite efficient nature-inspired search technique for global optimization. Since FA was developed, it has attracted a lot of attentions and becomes more popular in solving various real-world problems. FA is a swarm-based intelligence algorithm, which mimics the flashing behavior of fireflies [16]. A firefly flashes as a signal to attract others for some purposes, e.g. predating or mating. Accordingly, this biological phenomenon is formulated as a meta-heuristic algorithm depending on following three rules [16]:

- One firefly will be attracted to other fireflies regardless of their sex;
- The attractiveness is proportional to the brightness and decrease as their distance increases. For any two flashing fireflies, the less bright one will move towards the brighter one;
- The brightness of a firefly is determined by the landscape of the objective function.

Algorithm 2 Firefly Algorithm

```
Objective function f(t),  t = (ζ_1, ..., ζ_d)^T
Generate initial pop. P of fireflies
Light intensity I_i = f(ζ_i)
Define light absorption coefficient γ
while (t < MaxGeneration) do
    for each ζ_i ∈ P do
        for each ζ_j ∈ P do
            if (I_i < I_j) then Move ζ_i towards ζ_j end if
            Vary β with distance r via exp[−γr]
            Evaluate solutions and update light intensity
        end for j
    end for i
    Rank fireflies and find the current global best
end while
```

The pseudocode shown in Algorithm 2 summarizes the basic steps of the FA which consists of the three addressed rules.

4.1 Classic Firefly Algorithm

In the firefly algorithm, there are two important issues: the variation of light intensity and formulation of the attractiveness. For simplicity, we can always assume the attractiveness of a firefly is determined by its brightness, which in turn is associated with the encoded objective function.

The attractiveness function $\beta(r)$ can be any monotonically decreasing functions such as the following generalized form

$$\beta(r) = \beta_0 e^{-\gamma r^m}, \quad m \geqslant 1, \tag{8}$$

where β_0 is the attractiveness at $r = 0$, and r is the distance between two fireflies. As it is often faster to calculate $1/(1+r^2)$ than an exponential function [16], the Equation (8) can be approximated as

$$\beta(r) = \frac{\beta_0}{1 + \gamma r^2}. \tag{9}$$

The distance between any two fireflies i and j, at position x_i and x_j, respectively can be defined as a Cartesian distance:

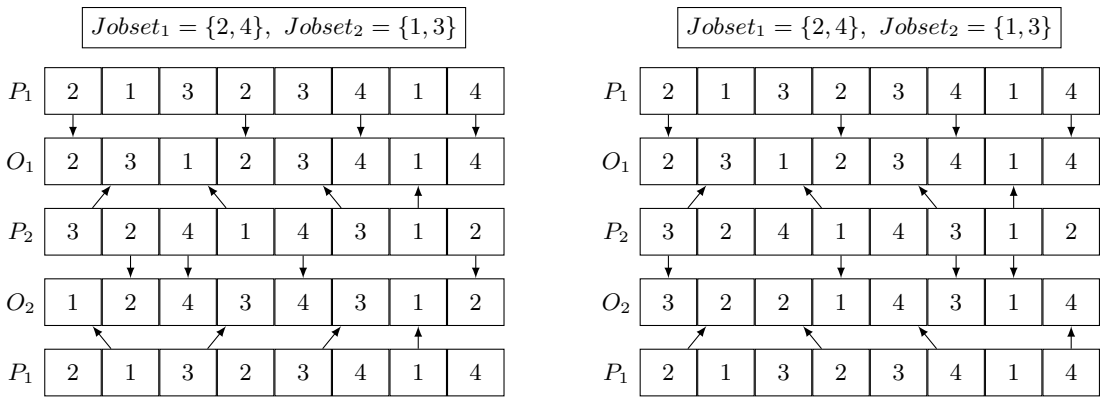
$$r_{\zeta_i \zeta_j} = \|x_{\zeta_i} - x_{\zeta_j}\| = \sqrt{\sum\nolimits_{k=1}^{d} (x_{\zeta_i k} - x_{\zeta_j k})^2}, \tag{10}$$

where $x_{\zeta_i k}$ is the kth component of the spatial coordinate x of the ζ_ith firefly.

The random movement of a firefly i towards another more brighter firefly j is determined by

$$x_{\zeta_i} = x_{\zeta_i} + \beta_0 e^{-\gamma r^2_{\zeta_i \zeta_j}} (x_{\zeta_i} - x_{\zeta_j}) + \alpha\, \epsilon_{\zeta_i}, \tag{11}$$

where the second term considers a firefly's attractiveness, the third term is randomization with α being the randomization parameter, and ϵ_i is a vector of random numbers drawn from a Gaussian distribution or uniform distribution. In a simplest form, ϵ_i can be replaced by $x - 1/2$, where x is a random number uniformly distributed in [0,1]. For most applications we can take $\beta_0 = 1$ and $\alpha \in [0, 1]$.



(a) POX crossover **(b) JBX crossover**

Figure 3: Crossover operators for OS string.

4.2 Discrete Firefly Algorithm for the FJSP

The FA has been originally developed for solving continuous optimization problems and cannot be directly applied to solve discrete optimization problems. The main challenges for using the FA to solve combinatorial optimization problems are the calculation of the discrete distance between two fireflies, and how they move in the coordination.

4.3 Distance

The discrete distance between two fireflies is defined by the distance between the permutation of its strings. The discretization of the distance, assimilation and revolution operators are described in details below.

4.4 Machine Assigment Distance

To calculate the distance between two MS strings, first, we apply the Hamming distance to verify positions (indexes) of the non-corresponding elements (values) of the strings, i.e. if $\zeta_{i1e} \neq \zeta_{j1e}$, $\forall e \in V$, then the index V_e is store in H_{ij}, $\{V_e \in H_{ij} : \zeta_{i1e} \neq \zeta_{j1e}\}$, where V_e is the eth element in V. Afterwards, these positions are used to calculate the MS distance as

$$\underline{d}_{ij} = \sum\nolimits_{\forall e \in H_{ij}} \left| m(\zeta_{i1e}, F(V_e)) - m(\zeta_{j1e}, F(V_e)) \right|, \quad (12)$$

where m is a function that maps the index of the machine ζ_{i1e} in the set F of the operation V_e. To illustrate Equation (12), consider strings $\zeta_{1,1} = \{5,3,1\}$ and $\zeta_{2,1} = \{1,3,4\}$, therefore $H_{1,2} = \{1,3\}$. Supposing that $F(V_1) = F(V_3) = \{1,3,4,5\}$, therefore $\underline{d}_{1,2} = |4-1| + \left|(1-3)\right|$.

4.5 Operation Sequence Distance

To determine the distance between any two OS strings we apply the so-called Swapping distance. The Swapping distance is the minimal number of swaps needed to perform in a string in order to produce another. In our implementation, every swap is considered a pair, where the first and second values in each pair designate the positions of the elements required to be swapped in order to advance one string towards another. The minimal number of swaps in order to move ζ_{i2} towards ζ_{j2} is stored in pair-wise fashion in S_{ij}. Following the calculation of the minimal amount of swaps, the distance between two OS strings can be determined as

$$\overline{d}_{ij} = \sum\nolimits_{\forall p \in S_{ij}} |p_1 - p_2|, \quad (13)$$

where p_1 and p_2 are respectively the first and second value of the pth pair. To illustrate Equation (13), consider $\zeta_{1,2} = \{2\ 3\ 1\}$ and $\zeta_{2,2} = \{1\ 3\ 2\}$, $S_{1,2} = \{(1,3)\}$, where $(1,3)$ means that ζ_{i21} should be swapped with ζ_{i23}, therefore $\overline{d}_{1,2} = 2$.

4.6 Moving Towards Another Firefly

In this study we split the movement into two sub-steps: assimilation β-step and revolution α-step. The steps β and α are not interchangeable, thereby, β-step must be computed before α-step while finding the new position. Both steps are described in details in Section 4.7 and Section 4.8. Both steps are illustrated in details on Table 2, where the firefly ζ_i updates its position towards a brighter firefly ζ_j. The parameters used in this illustration are $\beta_0 = 1, \gamma = 0.1$. The machines, jobs, operations, and topology considered in this illustration are based on the values shown in Table 1.

The β-step brings the iterated firefly closer to the brighter firefly by moving the MS string and the OS string towards the strings of the brighter firefly. Sections 4.7 and 4.8 explains in details the β-step. The α-step allows shifting the permutation into one of the neighboring permutations. Section 4.9 explains the α-step in details.

4.7 Moving the MS String

For every element e in H_{ij} a random number $x \in \underline{X}$ is generated uniformly distributed over $[0,1]$, where $\underline{X}_e$ is the eth random number for eth element of H_{ij}. In the case that $\underline{X}_e \leq \beta(\underline{d}_{ij})$, the distance of the element ζ_{i1e} respective to ζ_{j1e} is decreased by

$$floor\Big((\beta(\underline{d}_{ij}) \times \left|m\big(\zeta_{i1e}, F(V_e)\big) - m\big(\zeta_{j1e}, F(V_e)\big)\right|) + 1\Big). \quad (14)$$

4.8 Moving the OS String

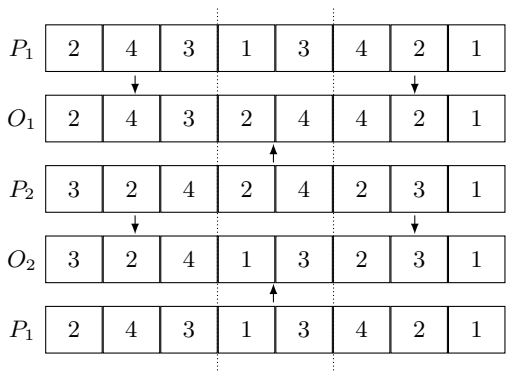


Figure 4: Two-point crossover.

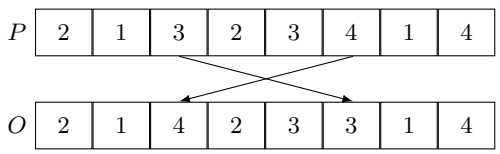


Figure 5: Swapping mutation.

For every pair p in S_{ij} a random number $x \in \overline{X}$ is generated uniformly distributed over $[0, 1]$, where $\overline{X}_p$ is the pth random number for pth pair of S_{ij}. In case that $\overline{X}_p \leq \beta(\overline{d}_{ij})$, the value in ζ_{i2p_1} is swapped with the value in ζ_{i2p_2}.

4.9 Random Movement

The α-step is performed on both strings as follows: (1) an element $e \in \zeta_{i1}$ is chosen at random and a new machine $k \in F(V_e)$ is assigned at random; (2) elements $e \in \zeta_{i2}$ and $e' \in \zeta_{i2}$ are chosen at random and swapped.

5. GENETIC ALGORITHM

Genetic algorithm (GA) is a class of algorithms based on the abstraction of Darwinian evolution of biological systems. Starting from an initial population, the algorithm applies genetic operators in order to produce offspring. At each generation, every new individual corresponds to a solution, i.e., a schedule of the given FJSP instance. The overall structure of our GA can be described as follows:

- *Coding:* the genes of the chromosomes describe the assignment of operations to the machines, and the operations sequences, as presented in Section 3.
- *Initial population:* the initial chromosomes are obtained by random permutation of jobs and machines.
- *Fitness evaluation:* the makespan is computed for each individual in the current generation as described in Section 3.4.
- *Selection:* the selection procedure is described in Section 5.1.
- *Offspring generation:* the new generation is obtained by changing the assignment of the operations and by changing the sequencing of operations. Each string is subjected to its own genetic operators. The crossover and mutation operators are discussed in Sections 5.2 and 5.5.
- *Stop criterion:* fixed number of generations is reached.

Table 3: Experiments with the YFJS instances. These instances are composed of Y-jobs.

Instance	Size	BRG		GA				FA			
		mks	CPU	*mks*	Mean	StDev	CPU	*mks*	Mean	StDev	CPU
YFJS01	4, 10, 7	773	11.5	773	773	0.0	3.98	773	773.0	0.0	2.16
YFJS02	4, 10, 7	825	9.88	825	825	0.0	6.49	825	825.0	0.0	2.89
YFJS03	6, 4, 7	347	3.72	347	348.8	3.6000	7.19	347	347.0	0.0	4.59
YFJS04	7, 4, 7	390	7.82	390	390	0.0	7.64	390	390.0	0.0	4.97
YFJS05	8, 4, 7	445	357.55	445	447.8	3.4292	12.87	445	447.1	3.2078	5.49
YFJS06	9, 4, 7	[425.29;449]	3600	447	447.5	1.0246	13.94	447	447.0	0.0	6.44
YFJS07	9, 4, 7	444	1392	444	445.2	3.6000	13.19	444	444.0	0.0	5.49
YFJS08	9, 4, 12	353	0.67	353	356.3	5.0408	11.49	353	353.0	0.0	4.88
YFJS09	9, 4, 12	242	14.03	242	243.8	2.7495	17.95	242	242.0	0.0	7.56
YFJS10	10, 4, 12	399	4.03	399	399.0	0.0	12.31	399	399.0	0.0	5.94
YFJS11	10, 5, 10	526	177.43	526	527.2	1.4696	37.56	526	526.9	1.3747	21.55
YFJS12	10, 5, 10	512	3218.89	512	512.4	0.4898	44.11	512	512.4	0.4898	28.74
YFJS13	10, 5, 10	405	1624.66	405	405.4	0.8000	174.12	405	405.2	0.6000	69.98
YFJS14	13, 17, 26	1317	3293.58	1317	1317.4	0.4898	186.22	1317	1317.2	0.4000	71.56
YFJS15	13, 17, 26	[1239;1244]	3600	1242	1246.6	6.2000	158.38	1239	1243.5	6.4691	75.23
YFJS16	13, 17, 26	[1200;1245]	3600	1222	1233.6	8.2849	135.32	1222	1225.2	2.1354	56.97
YFJS17	17, 17, 26	[1133;2379]	3600	1139	1141.5	4.2720	249.82	1134	1135.8	2.2271	105.32
YFJS18	17, 17, 26	[1220;2082]	3600	1230	1281.1	23.0844	305.13	1226	1237.2	10.9526	129.45
YFJS19	17, 17, 26	[926;1581]	3600	948	959.5	7.7103	326.79	943	952.3	6.8709	149.62
YFJS20	17, 17, 26	[968;2312]	3600	1001	1010.4	10.3169	311.33	974	978.0	2.0000	127.61

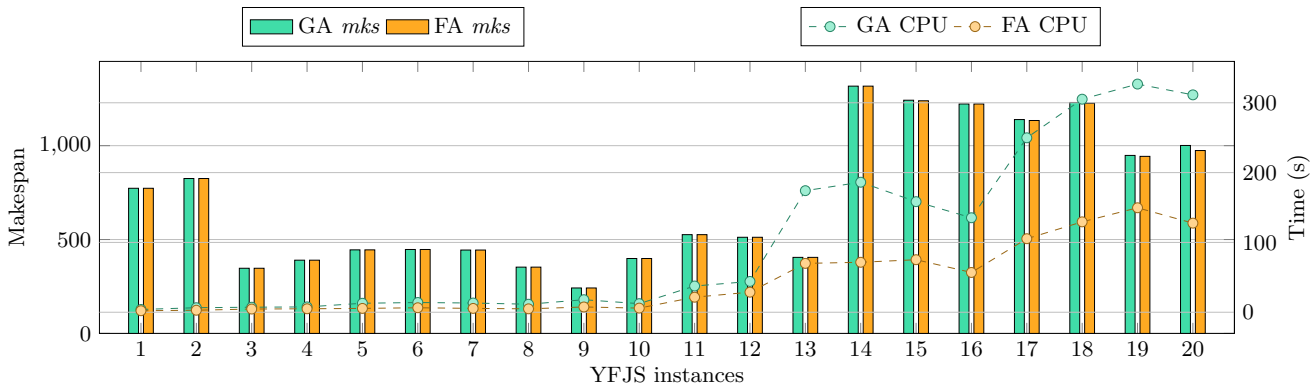


Figure 6: Best makespan and mean CPU time for the experiment with the YFJS instances.

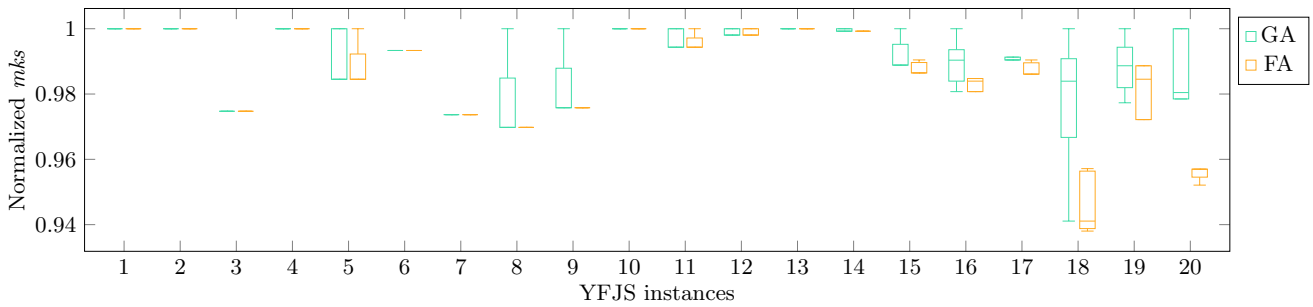


Figure 7: Box plot of the makespan obtained with the experiments involving the YFJS instances.

5.1 Selection

In GAs, the selection operator is used to select the individuals according to their fitness and maintain the highest quality chromosomes and characteristics within the population. In our algorithm, the selection strategy includes two parts: the method of keeping the best individuals and tournament selection. The method of keeping the best individuals is to copy the 1% of the best individuals for the next generation. The tournament selection strategy, proposed in [9], works as follows: two solutions are selected randomly as the parent solutions, if a random number generated between 0 and 1 is smaller than the probability r which usually is set to 0.8, then the better one is selected; otherwise, the worst one is selected.

5.2 Crossover

Crossover is the recombination of two parent chromosomes through the exchange of a part of one chromosome with a corresponding part of another in order to produce offspring. We denote the crossover probability as p_c. In this paper, two crossover operators: (a) precedence operation crossover (POX), proposed in [18]; (b) job-based crossover (JBX), proposed in [19]; are adopted for the OS string. During the OS string crossover procedure, one crossover operator is selected randomly.

5.3 Operation Sequence Crossover

The basic working procedure of POX is described as bellow (two parents are denoted as P_1 and P_2; two offspring are denoted as O_1 and O_2). Figure 3 shows an example of POX crossover operator.

1. The Job set $J = \{J_1, J_2, J_3, ..., J_n\}$ is divided into two groups $Jobset_1$ and $Jobset_2$ randomly;

2. Any element in P_1 which belongs to $Jobset_1$ are appended to the same position in O_1 and deleted in P_1; any element in P_2 which belongs to $Jobset_1$ are ap-

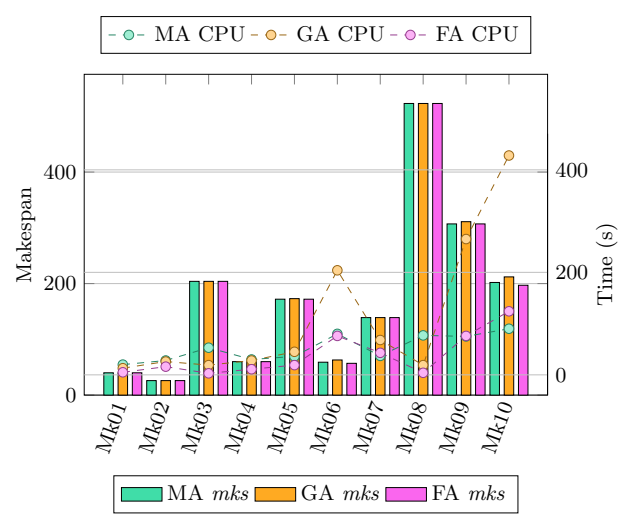


Figure 8: Experiment with FJSP instances.

Table 4: Experiment with the DAFJS instances. These instances are composed of G-jobs.

Instance	Size	BRG		GA				FA			
		mks	CPU	*mks*	Mean	StDev	CPU	*mks*	Mean	StDev	CPU
DAFJS01	4, 5-9,5	257	78.93	257	257.3	0.4714	10.33	257	257.0	0.0	4.95
DAFJS02	4, 5-7, 5	289	1271.7	289	290.7	1.1235	12.87	289	289.5	0.8844	5.46
DAFJS03	4, 10-17, 10	576	15.8	576	582.6	13.2	15.73	576	576.0	0.0	10.53
DAFJS04	4, 9-14, 10	606	1.22	606	606.0	0.0	31.55	606	606.0	0.0	9.17
DAFJS05	6, 5-13, 5	[347.53;403]	3600	395	398.5	2.7293	29.75	389	395.2	4.7638	7.23
DAFJS06	6, 5-13, 5	[326;435]	3600	416	421.3	3.2139	33.49	412	417.0	3.2455	6.16
DAFJS07	6, 7-23, 10	[497;562]	3600	519	524.7	4.0277	45.32	512	517.9	4.2968	9.43
DAFJS08	6, 6-23, 10	[628;631]	3600	628	633.1	5.1623	40.42	628	632.1	3.660	17.51
DAFJS09	8, 4-9, 5	[315;475]	3600	470	473.6	3.4020	35.16	464	469.4	1.9253	11.97
DAFJS10	8, 4-11, 5	[336;575]	3600	534	542.0	3.8471	49.62	533	538.2	5.2051	20.15
DAFJS11	8, 10-23, 10	[658;708]	3600	659	664.5	4.4252	110.64	659	662.1	2.205	55.43
DAFJS12	8, 9-22, 10	[530;720]	3600	652	656.6	4.0792	135.64	645	651.1	3.9643	66.48
DAFJS13	10, 5-11, 5	[304;718]	3600	662	666.5	3.5188	106.73	656	660.4	3.3823	46.98
DAFJS14	10, 4-10, 5	[358.95;860]	3600	736	748.2	9.4671	115.39	735	741.7	6.0955	51.97
DAFJS15	10, 8-19, 10	[512;818]	3600	677	686.8	5.7643	155.61	671	681.5	6.1738	71.95
DAFJS16	10, 6-20, 10	[640;819]	3600	679	688.9	7.9872	125.33	679	684.1	4.2719	61.73
DAFJS17	12, 4-11, 5	[300;909]	3600	804	813.4	6.2268	151.13	800	806.7	4.5821	59.49
DAFJS18	12, 5-9, 5	[322;951]	3600	803	808.2	4.3543	119.64	799	803.3	4.7703	40.95
DAFJS19	8, 7-13, 7	[512;592]	3600	525	527.5	2.3343	103.43	524	525.4	1.9596	49.27
DAFJS20	10, 6-17, 7	[434;815]	3600	712	720.9	5.3971	119.78	705	710.3	3.4538	69.72
DAFJS21	12, 5-16, 7	[504;965]	3600	815	819.9	5.1441	164.85	808	811.7	2.5157	75.81
DAFJS22	12, 5-17, 7	[464;902]	3600	720	724.5	5.3275	168.53	708	713.9	3.2958	85.49
DAFJS23	8, 6-17, 9	[450;538]	3600	476	480.5	2.9635	109.71	476	479.9	2.6297	61.51
DAFJS24	8, 6-25, 9	[476;666]	3600	568	572.8	3.4871	135.13	564	567.2	2.8566	55.32
DAFJS25	10, 9-19, 9	[584;897]	3600	755	758.5	3.5752	119.55	752	757.7	4.8397	69.84
DAFJS26	10, 8-17, 9	[565;903]	3600	751	756.9	6.1413	188.64	745	748.6	2.9394	78.81
DAFJS27	12, 7-22, 9	[503;981]	3600	838	845.1	6.9557	193.56	831	835.0	2.7568	81.55
DAFJS28	8, 8-15, 10	[535;671]	3600	545	552.9	4.3889	129.91	543	547.2	2.4000	60.48
DAFJS29	8, 7-19, 10	[609;726]	3600	657	663.7	4.2026	131.14	654	656.3	2.4129	61.41
DAFJS30	10, 8-19, 10	[467;656]	3600	557	564.9	5.5554	137.11	555	558.3	1.8856	65.36

pended to the same position in O_2 and deleted in P_2;

3. the remaining elements in P_2 are appended to the remaining empty positions in O_1 seriatim; and the remaining elements in P_1 are appended to the remaining empty positions in O_2 seriatim.

The second crossover operator for OS string is the job-based crossover (JBX). The basic working procedure of JBX is described below. Figure 3 shows an example of JBX crossover operator.

1. The Job set $J = \{J_1, J_2, J_3, ..., J_n\}$ is divided into two groups $Jobset_1$ and $Jobset_2$ randomly;

2. Any element in P_1 which belongs to $Jobset_1$ are appended to the same position in O_1; any element in P_2 which belongs to $Jobset_2$ are appended to the same position in O_2;

3. Any element in P_2 which belongs to $Jobset_2$ are appended to the remaining empty positions in O_1 seriatim; and any element in P_1 which belongs to $Jobset_1$ are appended to the remaining empty positions in O_2 seriatim.

5.4 Machine Selection Crossover

For the MS string, a two-point crossover has been adopted as the crossover operation. In this operation, two positions are selected at random. Based on the selected positions, two children strings are created by swapping all elements between the positions of the two parent strings. Figure 4 shows an example of two-point crossover.

5.5 Mutation

Mutation operator is applied on a single chromosome for the purpose of changing a gene at its respective location. The gene 1011 can be mutated as 1111, as the gene at location 2 is flipped from 0 to 1. The mutation operator is used to change some information in a selected chromosome or diversify the solution space for further exploration. The

Table 5: Parameters of the GA and FA.

Instance	F/I	$T(fa)/T(ga)$	γ	p_c	p_m
Small	100/250	400/300	0.10	0.85	0.02
Medium	250/500	800/500	0.10	0.85	0.02
Large	500/750	1200/750	0.10	0.85	0.02

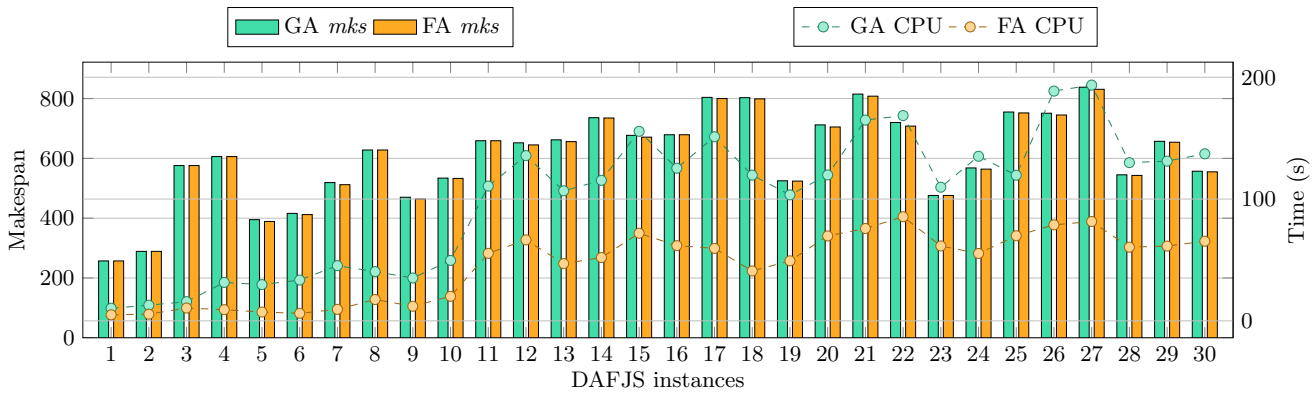


Figure 9: Best makespan and mean CPU time for the experiment with the DAFJS instances.

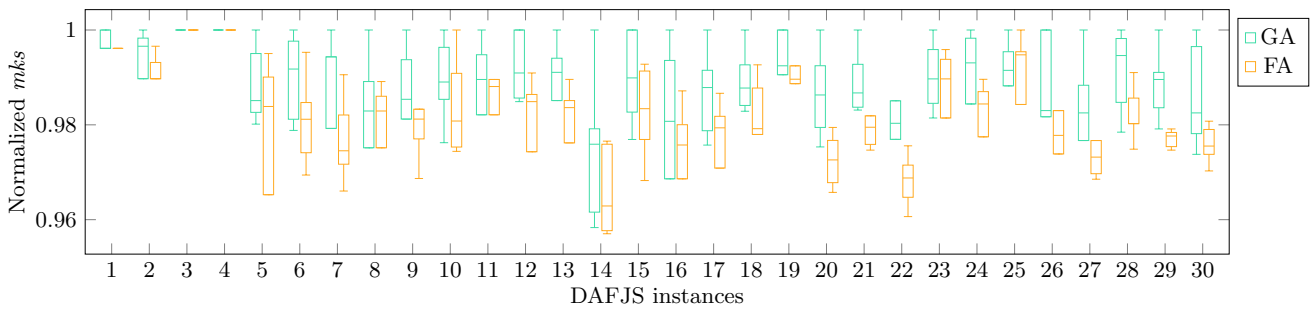


Figure 10: Box plot of the makespan obtained with the experiments involving the DAFJS instances.

mutation probability p_c should be small since a high probability will be adverse for the information preservation of the good chromosomes. In this paper, the swapping mutation [19], has been adopted for the OS string. In this procedure, two positions are selected and its respective elements are swapped. A single point mutation is used for the MS string. In this procedure, a position of the MS string is selected, and a new machine is assigned for its respective operation. Figure 5 shows an example of swapping mutation.

6. NUMERICAL RESULTS

This section presents the results of computational experiments involving the algorithms. We used as benchmarks 10 FJSP instances (named Mk) introduced in [2], 20 EFJSP instances composed of Y-jobs (named YFJS) proposed in [1], and 30 EFJSP instances composed of G-jobs (named DAFJS) further proposed in [1]. Experiments involving the metaheuristics were conducted on the same computer, an Intel Core i7 2.70GHz. The computer used for the MILP model has an Intel Xeon E5440 2.83GHz processor. The proposed algorithms were implemented in C++. A supplementary material (e.g., the instances used in this work) is provided at `https://willtl.github.io/acr2018_summer/`.

Based on experiments we find out the best parameters for the GA and FA based on the size of the instances. We divided into three categories (i.e., small, medium, large), and for each category. Small instances, i.e., less or equal to 6 jobs and 5 machines; medium instances, i.e., less or equal to 10 jobs and 8 machines; larger instances, i.e., instances that does not belong to another group. The number of individuals, fireflies, and the number of generations are shown in Table 5, where F is the number of fireflies, I is the number of individuals, T is a function that gives the number of generations for each algorithm. The attraction coefficient was kept as $\beta_0 = 1.0$ for all experiments.

Figure 8 presents the results for the FJSP instances proposed in [2]. The bars denote the *mks* and the lines represent the CPU time, in seconds. This set of instances include only path-jobs. We analyze our proposed methods with another state-of-the-art algorithm, a memetic algorithm (MA) proposed in [17]. The MA was implemented on an Intel Core i7-3520M 2.9 GHz processor in Java.

We experiment our algorithms with two sets of EFJSP instances proposed in [1]. Figure 6 and Figure 9 shows the best *mks* and CPU time for the experiments with the YFJS and DAFJS instances. Figure 7 and Figure 10 shows the variation of observed data through quartiles for the experiments with the YFJSP and DAFJS instances. Tables 3 and 4 present the numerical results for the experiments involving the YFJSP and DAFJS instances. The instance column designates the names of the instances. For the MILP model [1] (we name it BRG), the *mks* column designates the optimal makespan or the lower and upper bounds found by CPLEX. The CPU column records the CPU time in seconds. The CPLEX was limited to 3600 seconds. Regarding the metaheuristics, for each instance, the experiment was performed 25 times. Therefore, column *mks* designates the best-obtained makespan, column Mean records the mean of

the best-obtained values, column StDev records the standard deviation, and the column CPU designates the mean CPU times.

On the experiment among the [2] instances, we can perceive that the FA was able to find the optimal solution for most of the instances except the two largest instances (i.e. Mk06 and Mk10). The FA is more efficient related to the GA and MA, however, the MA was more efficient for the MK10. Based on the investigations with Y-job and G-job instances given on Tables 3 and 4, we can see that both the GA and FA loses efficiency for larger instances due to the fact that the parameters had to be adjusted in order to increase the searching capabilities of the algorithms. In that case, even with readjusted parameters (e.g., a higher number of individuals, fireflies, and iterations), the GA was not able to obtain similar solutions to those that FA found for the largest instances. The MILP model is more efficient for smaller instances, and as expected, it does not scale well for larger instances.

Here is a summary of our results. The proposed FA is more effective and efficient than the proposed GA. As expected, both the GA and FA are more efficient than the MILP. Based on the bounds given by the MILP model, we can see that the FA decreased the gap (i.e. the distance between lower and upper bounds) for several instances.

7. CONCLUSION

In the present paper, we extended the definition of the FJSP taking into account parallel operations in the route of the jobs. We put forward a discrete firefly algorithm and a genetic algorithm to solve the EFJSP. In order to evaluate the performance of the solution methods, 50 EFJSP instances were used in the computational experiments. Furthermore, 10 famous FJSP instances were used to provide comparisons with other state-of-the-art algorithms. To evaluate the performance of the proposed algorithms, a MILP model was used to provide optimal solutions or solution bounds. The experiments with the GA and FA shows that both are possible approach for the considered problem.

8. REFERENCES

[1] E. G. Birgin, P. Feofiloff, C. G. Fernandes, E. L. De Melo, M. T. Oshiro, and D. P. Ronconi. A milp model for an extended version of the flexible job shop problem. *Optimization Letters*, 8(4):1417–1431, 2014.

[2] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

[3] I. A. Chaudhry and A. A. Khan. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591, 2016.

[4] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[5] Y. Demir and S. K. İşleyen. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*, 37(3):977–988, 2013.

[6] P. Fattahi, M. S. Mehrabad, and F. Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of intelligent manufacturing*, 18(3):331, 2007.

[7] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert systems with applications*, 42(21):7652–7663, 2015.

[8] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[9] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

[10] E. L. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in operations research and management science*, 4:445–522, 1993.

[11] X. Li and L. Gao. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, 174:93–110, 2016.

[12] M. Lohrer. *A comparison between the firefly algorithm and particle swarm optimization*. PhD thesis, 2013.

[13] W. T. Lunardi, L. H. Cherri, and H. Voos. A mathematical model and a firefly algorithm for an extended flexible job shop problem with availability constraints. In *International Conference on Artificial Intelligence and Soft Computing*, pages 548–560. Springer, 2018.

[14] W. T. Lunardi and H. Voos. Comparative study of genetic and discrete firefly algorithm for combinatorial optimization. In *33rd Symposium On Applied Computing (SAC), 2018*, pages 1–8. ACM/SIGAPP, 2018.

[15] G. Vilcot and J.-C. Billaut. A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*, 190(2):398–411, 2008.

[16] X.-S. Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[17] Y. Yuan and H. Xu. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12(1):336–353, 2015.

[18] C. Zhang, P. Li, Y. Rao, and S. Li. A new hybrid ga/sa algorithm for the job shop scheduling problem. *Evolutionary computation in combinatorial optimization*, pages 246–259, 2005.

[19] G. Zhang, L. Gao, and Y. Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38(4):3563–3573, 2011.

ABOUT THE AUTHORS:

Willian Tessaro Lunardi received his M.S. degree in computer science from the Pontifical Catholic University of Rio Grande do Sul, Brazil, in 2016. Currently, he is a Ph.D. candidate at the University of Luxembourg, Automation & Robotics Research Group. His current research interests include optimization techniques, approximate algorithms, planning and scheduling, and metaheuristics.

Holger Voos received his Ph.D. degree in electrical engineering from the University of Kaiserslautern, Germany, in 2000. Currently, he is a professor at the University of Luxembourg and head of the Automatic Control Laboratory. From 2004 to 2010, he served as a professor at the University of Applied Sciences Ravensburg-Weingarten, Germany. He was the head of the Laboratory of Mechatronics and Mobile Robotics, chairman of the international Master program mechatronics, and a member of the Board of Governors. From 2000 to 2004, he worked as a systems engineer and a project manager in the research and development department at Bodenseewerk Geretechnik GmbHerlingen, Germany (now: Diehl-BGT-Defence GmbH). His current research interests include optimization techniques, distributed and networked control, autonomous robots, safety, and mechatronic systems.