RESOURCES ALLOCATION IN INDOOR AND UNDERWATER VLC NETWORKS

BY

KHALED ABDUL-AZIZ AL-UTAIBI

A Dissertation Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE ENGINEERING

KING FAHD UNIVERSITY
OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

APRIL, 2019

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by **KHALED ABDUL-AZIZ AL-UTAIBI** under the direction of his dissertation adviser and approved by his dissertation committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE ENGINEERING**.

Dissertation Committee

Dr. Sadiq M. Sait (Adviser)

Dr. Saad Al-Ahmadi (Member)

Dr. Aiman El-Maleh (Member)

Dr. Moataz Ahmad (Member)

Dr. Murat Uysal (Member)

Dr. Ahmad Al-Mulhem
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

Date

To the memory of my father and to my mother.

ACKNOWLEDGMENTS

Acknowledgement due to King Fahd University of Petroleum & Minerals for supporting this research. I would like to express my appreciation to my dissertation advisor Dr. Sadiq Sait Mohammed for his guidance, patience, and sincere advice throughout this work. I acknowledge him for his valuable time, constructive criticism, and stimulating discussions. Thanks are also due to my dissertation committee members, Dr. Saad Al-Ahmadi, Dr. Aiman El-Maleh, Dr. Moataz Ahmad, and Dr. Murat Uysal for their comments and critical review of the dissertation. I am very thankful to the Department Chairman Dr. Ahmad Al-Mulhem and department secretary for their cooperation and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A	Photo-detector Physical Area
D_R	Diameter of the Aperture Receiver
N_0	Noise Density
$\Phi_{1/2}$	Transmitter Semiangle
$\Psi_{1/2}$	Receiver FOV Semiangle
R	O/E Conversion Coefficient
$\theta_{1/e}$	Total Width Transmitter Beam Divergence Angle

DISSERTATION ABSTRACT

NAME:	Khaled Abdul-Aziz Al-Utaibi
TITLE OF STUDY:	Resources Allocation in Indoor and Underwater VLC Networks
MAJOR FIELD:	Computer Science Engineering
DATE OF DEGREE:	April, 2019

Visible light communication (VLC) refers to a new wireless communication technology that transmits information through light source (LED) intensity modulation at high rates. VLC systems are characterized by several benefits as opposed to RF-based wireless communications that include information security, energy efficiency, low cost, and broad spectra. As a revolutionary remedy to wireless communications with several applications, the VLC technology has attracted the attention of researchers on different levels such as modulation techniques, resource allocation, load balancing, and handover. In this dissertation, we address the issue of resource allocation by taking into account load aware dynamic LED allocation within a centralized light access network. We have proposed a new model for VLC resource allocation (RA) to optimize LEDs assignment to users using a Tetris game model. The objective is to maximize system data rate

under the considered partial fairness. We have developed two meta-heuristics based on this model to allocate LEDs to users in a centralized light access network. The algorithms are based on simulated annealing and simulated evolution. The performance of the proposed model is compared with a direct application of the simulated annealing (SA) algorithms in terms of achieved data rate and convergence rate.

ملخص الرسالة

الاسم: خالد بن عبدالعزيز العتيبي

عنوان الدراسة: تخصيص الموارد في شبكات الاتصالات الضوئية المرئية الداخلية وتحت سطح الماء

التخصص: علوم وهندسة الحاسب الآلي

تاريخ الدرجة العلمية: شعبان ١٤٤٠

يشير اتصال الإضاءة المرئية إلى تقنية اتصالات لاسلكية جديدة تنقل المعلومات من خلال تعديل كثافة مصدر الضوء بمعدلات عالية. تتميز أنظمة اتصال الإضاءة المرئية بالعديد من الفوائد على عكس الاتصالات اللاسلكية القائمة على تردد موجات الراديو، والتي تشمل أمن المعلومات، وكفاءة الطاقة، والتكلفة المنخفضة، والأطياف الواسعة. كعلاج ثوري للاتصالات اللاسلكية مع العديد من التطبيقات، جذبت تقنية اتصال الإضاءة المرئية انتباه الباحثين على مستويات مختلفة مثل تقنيات التشكيل، وتخصيص الموارد، وموازنة التحميل، والتسليم. في هذه الرسالة، نعالج مسألة تخصيص الموارد من خلال مراعاة تخصيص ديناميكي للمصابيح يراعي الأعباء داخل شبكة وصول الضوء المركزية. لقد اقترحنا نموذجًا جديدًا لتخصيص الموارد لتحسين يعتمد على نموذج لعبة الفيديو الشهيرة (تيترس) ويهدف هذا النموذج إلى زيادة معدل بيانات النظام إلى أقصى حد وفقًا للعدالة الجزئية. لقد قمنا بتطوير اثنين من الخوارزميات بناءً على هذا النموذج لتخصيص المصابيح للمستخدمين في شبكة وصول الضوء المركزية. وتستند الخوارزميات على خوارزمية محاكاة التلدين وخوارزمية محاكاة التطور. وقد تم مقارنة أداء النموذج المقترح مع التطبيق المباشر لخوارزمية محاكاة التلدين من حيث معدل البيانات المحقق ومعدل التقارب.

CHAPTER 1

BACKGROUND OF VLC SYSTEMS

The rising quantity of mobile gadgets alongside the global prominence characterizing multimedia and social media applications have resulted in an extensive escalation within mobile traffic [1]. According to Cisco Visual Network Index [2], traffic for mobile data has increased 7-fold in the last 5 years. Additionally, recent Cisco forecasts indicate that traffic for mobile data is anticipated to hit 49 exta-bytes every month by 2021 as shown in Fig.1.1.

Currently, digital wireless systems function within the RF band in less than 6 GHz. To overcome the huge wireless bandwidth demand, different solutions are being investigated and suggested, for instance, multi-input multi-output (MIMO), millimeter-wave, and ultra-densification [3]. Despite the limitations and challenges of such solutions, additional bandwidth is required to meet the anticipated growth in data traffic in the future [4]. The optical band, as illustrated in Fig. 1.2, provides a greater bandwidth

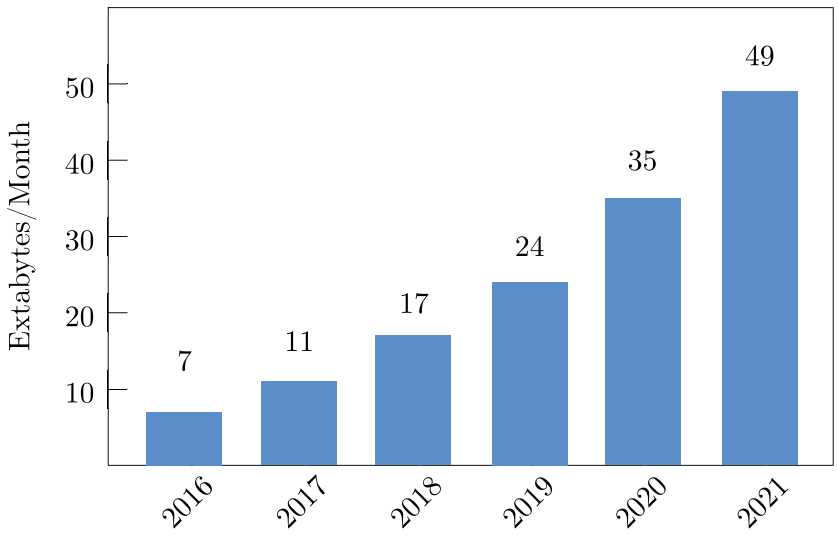

Figure 1.1: Global mobile data traffic, 2016 to 2021 (Source: Cisco VNI Mobile, 2017).

compared to the RF band through numerous magnitude order.

Visible light communication (VLC) [5] refers to a new technology, which can address the drawbacks associated with the wireless communication systems' spectrum [6]. The main concept behind VLC is the transmission of information through light source (LED) intensity modulation at high rates, which preserve the levels of illumination and not visible to the human eye.

VLC is characterized by several benefits than RF-based communication systems [7]:

1. Broad spectra: the spectra for visible light occupies the range of frequency from 400THz to 800 THz, thus making it a potential solution to overcoming the crowded RF spectra (3kHz to 300GHz) within wireless communication systems.

2. Safety: Because light does not present any electromagnetic interruption, VLC is ideal for communication in specific settings such as airplanes and hospitals.

3. Cheap: VLC systems could decrease the costs of wireless communication systems through current lighting infrastructure containing few additional data transmission modules.

4. High-energy efficiency: the main source of light (LEDs) for VLC systems has been found to decrease the consumption of energy by up to 80% as opposed to conventional sources of light.

5. Information security: Compared to RF communication that is capable of penetrating walls, triggering information leaks, VLC systems could offer more secure communication connections as light cannot pass through opaque objects.

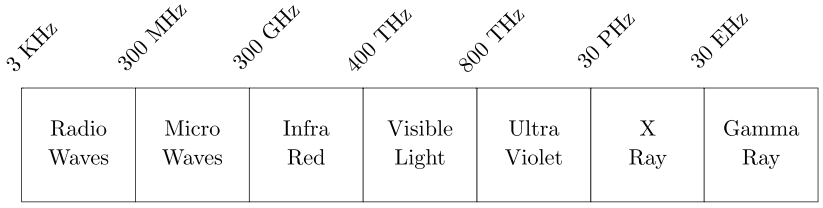

Figure 1.2: The electromagnetic spectrum.

The history for VLC can be traced to Alexander Graham Bell's discovery of the photohone during the early 1880s, where he was able to convey speech in modulated sunlight for hundreds of meters. Latest works on VLC commenced in 2003 at Nakagawa Laboratory (Keio University in Japan) through a source of light to convey data. From that time, there have been extensive research activities that focused upon VLC.

Fig. 1.3 illustrates a block diagram for a typical VLC system. LED is extensively utilized as the source of light, ejecting necessary optical signals. The photo-detector (receiver) gathers the additional light signal (optical noise). The optical noise reduces the signal quality, and hence the optical concentrator is utilized at the side of the receiver to enhance the signal-to-noise ratio.

Inherent characteristics of VLC systems, including information security, energy efficiency, low cost, safety, and broad spectra made them viable for communication in different areas including hospitals, vehicle-to-vehicle, underwater, and indoor . The focus of this proposal will be on underwater and indoor communications.

1.1 Indoor of VLC Systems

Fig.1.4 shows the indoor VLC (LiFi) network concept. Several LED light bulbs offer optical and illumination access points (APs) to the users within the room. An optical downlink is created between user's gadget and the AP through modulation of the LED bulb at high rates that cannot be seen by human eyes. An optical up-link is implemented through an array of transmitters on user gadgets and receivers adjacent to the AP. In most cases, IR is used as it is not visible to the users. The main functions

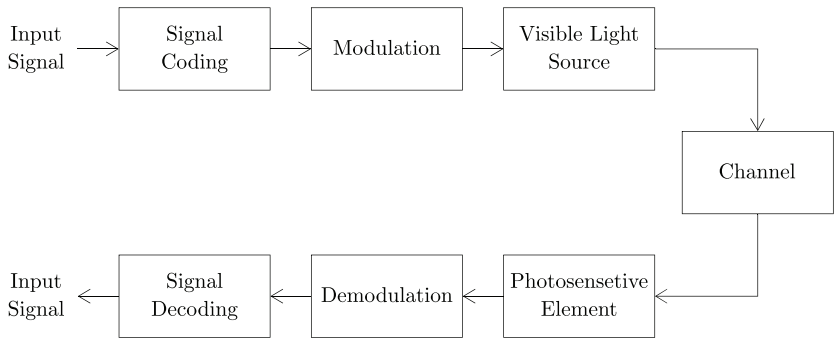

Figure 1.3: Block diagram of a typical visible light communication system.

of the central control unit include controlling the handover process from a single AP towards another when terminals are in motion, handling multi-user access, control interruptions between downlink channels, and assigning resources to users.

Just as WiFi, LiFi constitutes a wireless communication technology that utilizes similar 802.11 protocols; however, it utilizes VLC (instead of RF waves) that has larger bandwidth. Notably, IEEE has initiated the global standardization work via the IEEE 802.15.13 Task Force to come up with another VLC standard [8]. The standard presents a definition of media access control (MAC) and physical (PHY) layers for optical wireless communications. The standard can deliver rates of data of up to 10Gbit per second over distances in the range of 200m unlimited sightline. It is developed for point-to-point as well as point-to-multipoint communication in coordinated and non-coordinated topologies. The standard features adaptation to different channel conditions and retaining connectivity while shifting in the single coordinator range or shifting between coordinators.

1.1.1 Channel Modeling

Typically, LED-driven VLC systems undergo implementation with a line of sight (LOS) and intensity modulation and direct detection (IM/DD) scheme [9]. Fig.1.5 illustrates IM/DD receiver/transmitter gadgets for VLC systems. Within the transmitter, the drive current undergoes modulation through a modulating signal $m(t)$ that in turn varies the optical source intensity $x(t)$. The receiver uses a photodiode (photo-detector) to locate the optical signals and transform them into photo-current $y(t)$ that is utilized for recovering the information conveyed.

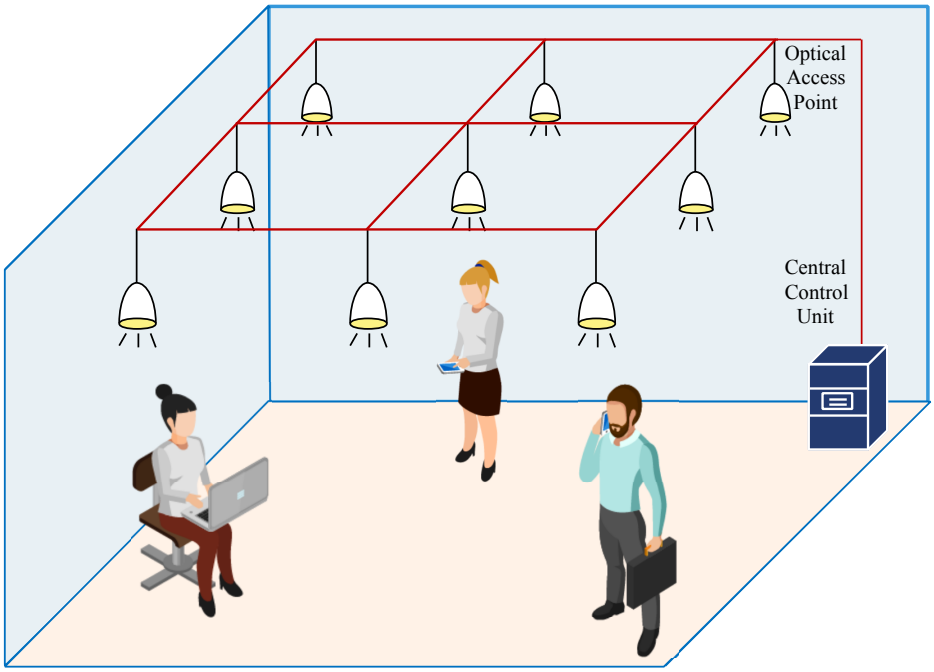

Figure 1.4: The concept of LiFi network.

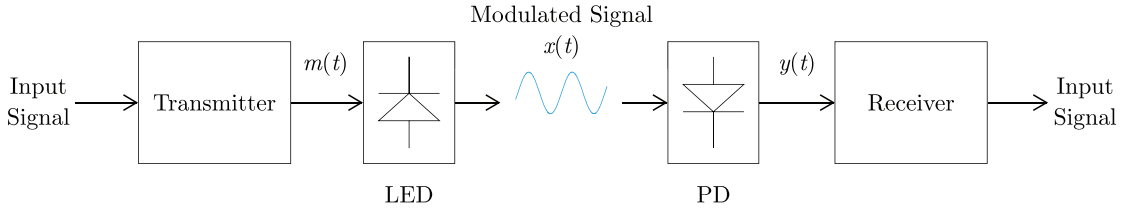

Figure 1.5: Block diagram of an intensity modulation with direct detection communication channel.

Fig.1.6 illustrates the VLC system' channel model utilizing IM/DD; the photo-current $y(t)$ is proportional to the instant received optical power incident on the photo-detector and is expressed through [10]:

$$\begin{aligned} y(t) &= Rx(t) \otimes h(t) + N_0(t) \\ &= \int_{-\infty}^{+\infty} Rx(\tau)h(t-\tau)d\tau + N_0(t) \end{aligned} \tag{1.1}$$

where $\otimes$ represents convolution, N_0 denotes the AGWN, $h(t)$ denotes the channel's impulse, $x(t)$ denotes the instant optical power radiated from LED, R denotes the responsivity of the detector, and $y(t)$ represents the photo-current received. Note that $x(t)$ represents a power signal which imposes two restrictions: (1) $x(t)$ must be positive, and (2) the average value of $x(t)$ must not exceed a specified maximum power value to satisfy eye safety requirements

The channel impulse response $h(t)$ could be modeled based on the following equation [11]:

$$h(t) = \begin{cases} \frac{2t_0}{t^3 \sin^2(FOV)} & t_0 \leq t \leq \frac{t_0}{\cos(FOV)} \\ 0 & \text{elsewhere} \end{cases} \tag{1.2}$$

where t_0 is the minimum delay and FOV is the field of view.

Considering the line-of-site (LOS) channel model illustrated in Fig.1.7, the channel

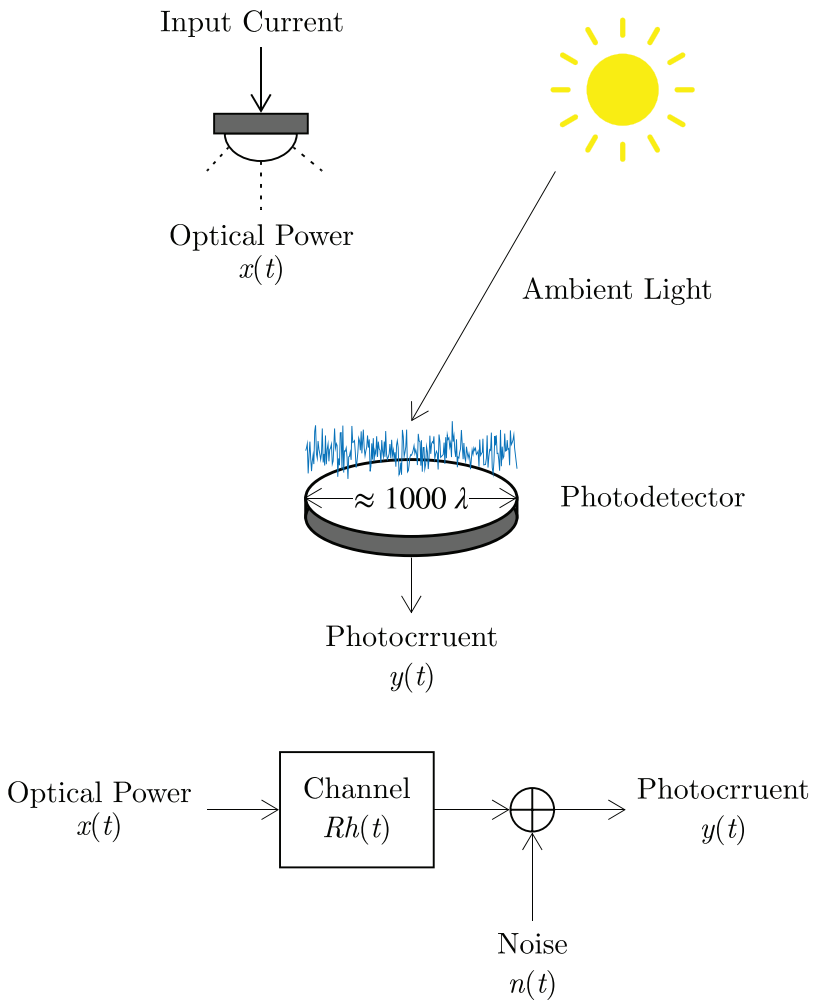

Figure 1.6: Equivalent baseband model of an optical wireless system using IM/DD.

DC gain for the receiver can be approximated by [10]:

$$H_{\text{los}}(0) = \begin{cases} \frac{(m+1)A}{2\pi d^2} \cos^m(\phi) \cos(\psi) & 0 \leq \psi \leq \Psi_{1/2} \\ 0 & \psi > \Psi_{1/2} \end{cases} \tag{1.3}$$

where A denotes the photo-detector physical area, d denotes the distance between the receiver and the transmitter, ϕ denotes the irradiance angle with regard to the transmitter perpendicular axis, ψ denotes the incidence angle with regard to the perpendicular axis of the receiver, whereas $\Psi_{1/2}$ denotes the FOV semi-angle concentrator.

The Lambertian order m is expressed by:

$$m = \frac{-\ln 2}{\ln(\cos(\Phi_{1/2}))} \tag{1.4}$$

Where $\Phi_{1/2}$ represents the LED half-power angle; for instance, $\Phi_{1/2} = 60^\circ$ corresponds to $m = 1$.

The relation between the channel DC gain and the channel impulse response is given by:

$$H(0) = \int_{-\infty}^{+\infty} h(t)dt \tag{1.5}$$

The optical power received P_r is expressed based on the transmitted power P_t and the channel gain as shown:

$$P_{r-\text{los}} = H(0)_{\text{los}} P_t \tag{1.6}$$

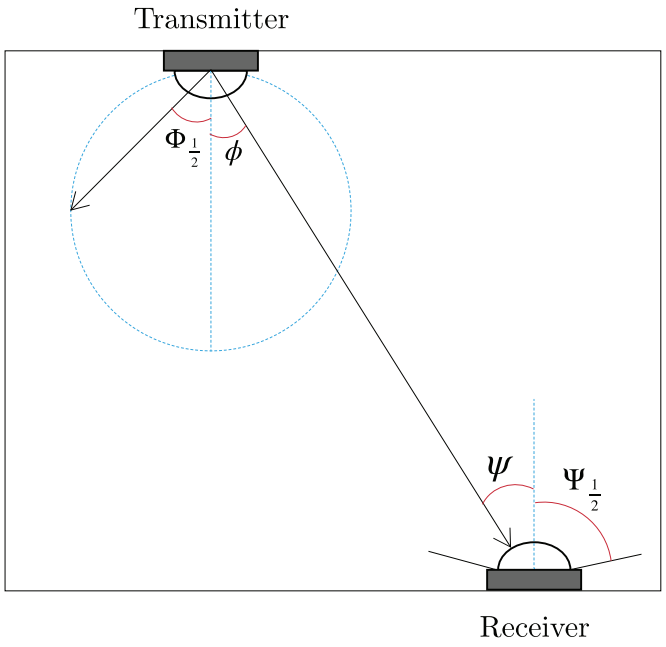

Figure 1.7: VLC downlink transmission geometry of a LOS path.

1.1.2 Path Loss Model

Within communication systems, path loss refers to the attenuation that electromagnetic waves experience in transit between a receiver and a transmitter. Path loss could be attributed to several effects that include absorption, aperture-media coupling loss, reflection, refraction, and loss of free space. In most cases, dB constitutes the units for expressing path loss. The channel's linear path loss could be written as the transmit power-receiver power ratio:

$$P_L = \frac{P_t}{P_r} = \frac{1}{H(0)} = (H(0))^{-1} \tag{1.7}$$

Expressing path loss in dB, we obtain:

$$P_L\text{dB} = 10\log_{10}(H(0))^{-1} = -10\log_{10}(H(0)) \tag{1.8}$$

1.2 Underwater VLC Systems

Demand for underwater systems of communication are on the rise because of the continuing human activity expansion in underwater settings, for instance, tactical surveillance, port security, offshore oil explorations, maritime archeology, underwater scientific collection of data, and environmental tracking [12]. Three technologies utilized for underwater communication exist namely optics, RF, and acoustics [13].

The acoustic technique constitutes the most extensively utilized technology within UWC because it is capable of achieving a prolonged connection ranging up to many kilometers [14]. Nevertheless, it is characterized by numerous technical pitfalls. Firstly, acoustic connections have low rates of data (in Kbps) [15]. Secondly, acoustic links are prone to serious communication delays (mainly in seconds) because of the low propagation velocity of sound waves within water (around 1500m/s for pure water at 200C. Therefore, they might not be utilized for real-time exchange of data. Thirdly, acoustic transceivers are often energy consuming, expensive and bulky [16].

RF waves could initiate smooth transition via water/air interfaces and have a higher tolerance for water turbulence [13]. Nevertheless, the main drawback of underwater RF communications is the short range of connection. The RF waves could solely propagate a few meters in additional –low frequencies (30-300Hz) [15]. Besides, the underwater RF systems necessitate energy-consuming, expensive, and bulky receivers and transmitters.

Unlike RF and acoustic underwater techniques, underwater visible light communication (UVLC) is characterized by the lowest costs for implementing it, lowest connection delay, highest security and highest rates of data transmission (in the sequence of Gbps on moderate distances) [13].

Because water is transparent to green or blue light (450 nm – 550 nm), light emitting diodes (LEDs) or visible light lasers could be utilized as transmitters of underwater wireless connectivity with rates of data in the sequence of 10 Mb/s. Experimental UVLC demonstrations that surpass Gb/s were found within a regulated laboratory setting [17]–[20]. The past few years have been characterized by increased literature

about VLC spanning from the modeling of channels to issues of the upper layer and physical layer [21]–[34].

1.2.1 Path Loss Model

The UVLC path loss constitutes the geometrical loss and attenuation loss function. The attenuation loss is identified through scattering and absorption. Absorption refers to an energy transfer process wherein photons forego their energy and transform it into other types, such as chemical or heat. Scattering is the light deflection from the initial path because of interactions with transmission media atoms and molecules.

The Beer-Lambert formula [35] is extensively utilized within the literature for computing underwater path loss.

$$PL_{BL}(d) = 10 \log_{10} e^{-cd} \tag{1.9}$$

where d represents the connection range between receivers and transmitters, whereas c denotes the overall attenuation. The coefficient c could be denoted as $c = a + b$, where a and b represent the scattering and absorption coefficients. The Beer-Lambert principle rides on two assumptions that underestimate the optical power received. Firstly, the receiver and transmitter have perfect alignment. Secondly, all scattered photons become displaced even though some scattered photons could still reach the receivers following several events of scattering.

Geometrical loss is caused by the dispersal of transmitted beams between the receiver and the transmitter. In most cases, the beam disperses to sizes that are

larger than the aperture of the receiver, thus leading to the loss of the overfill energy. Taking into account the line-of-sight (LOS) configuration alongside semi-collimated laser sources with Gaussian beam-shapes, the geometrical loss could be estimated as [36]:

$$PL_{GL}(d) \approx 10\log_{10}\left(\left(\frac{D_R}{\theta_{1/e}d}\right)^2\right) \tag{1.10}$$

where D_R represents the diameter of the aperture receiver whereas $\theta_{1/e}$ represents the total width transmitter beam divergence angle1. Consistent with (1.9) and (1.10), the total path loss is expressed by:

$$\begin{aligned} PL(d) &= PL_{BL}(d) + PL_{GL}(d) \\ &\approx 10\log_{10}\left(\left(\frac{D_R}{\theta_{1/e}d}\right)^2 e^{-cd}\right) \end{aligned} \tag{1.11}$$

Practically, the detector could receive rays following several multiple scattering while the formulation above rides on the assumption that scattered rays are unavailable. To consider the impetus of scattered rays, an improved version for (1.11) is suggested in [37]:

$$PL(d) \approx 10\log_{10}\left(\left(\frac{D_R}{\theta_{1/e}d}\right)^2 e^{-cd\left(\frac{D_R}{\theta_{1/e}d}\right)^T}\right) \tag{1.12}$$

where another term that is proportional to light source geometrical propagation is unveiled within the negative exponential. In equation (1.12), T represents a correction coefficient that could be identified by fitting data to simulated data.

1.2.2 Turbulence Channel Model

Most existing literature regarding underwater optical wireless communications [22], [30], [38], [39] take into account deterministic path loss models at the expense of turbulence-induced fading. Underwater optical turbulence takes place because of the seawater refractive index changes and unveils instantaneous reductions on the mean received power [13]. Ocean currents are the common cause of such a phenomenon that induces sudden differences in water pressure and temperature.

The underwater optical turbulence (UOT) model was borrowed from the classical log-normal turbulence model utilized within free-space optical (FSO) communication [40]:

$$f_I(I) = \frac{1}{\sqrt{2\pi}\sigma_t I}\exp\left(-\frac{(\ln I - \mu)^2}{2\sigma_t^2}\right), I > 0 \tag{1.13}$$

where $f_I(I)$ denotes the UOT PDF, I represents the obtained optical irradiance, μ represents the average logarithmic light density whereas σ_t^2 denotes the index of scintillation.

1.3 Modulation Techniques

Notably, four modulation schemes that are utilized in VLC communication exist; they include Color Shift Keying (CSK), Orthogonal Frequency Division Modulation (OFDM), Pulse Modulation Methods, and On-Off Keying (OOK). In [41], the performance for such schemes is evaluated based on power requirements, Bit Error Rate (BER), Signal to Noise Ratio (SNR) and data rate.

1.3.1 On-Off Keying (OOK)

Within the scheme, the source of light is switched on during transmission of logic 1 and is switched off during logic 0 transmission. In off state, the light intensity is reduced (not totally switched off). Fig. 1.8 illustrates the string of data "101101" with OOK-RZ (Return-to-Zero) and OOK-NRZ (Non-Return-to-Zero) modulation schemes. The OOK scheme provides appropriate characteristics, for instance, simple implementation. Nevertheless, the major OOK restriction is the lower rates of data, particularly in the maintenance of various levels of dimming.

1.3.2 Pulse Modulation Methods

Within the schemes, pulse position and pulse width represents the conveyed data. In Pulse Width Modulation (PWM) scheme, the conveyed signal undergoes modulation through digital pulses, whereby pulses that correspond to logic 1 possess varying width as opposed to those denoting logic 0. Within Pulse Position Modulation (PPM), the pulses width remains unchanged while each pulse's position is varied on the basis of the logical values earmarked for transmission. The Variable Pulse Position Modulation (VPPM) constitutes the PWM and PPM consolidation where pulse position and width undergo variation. Fig.1.9 presents a schematic diagram that shows the variation between VPPM, PPM, and PWM. Other pulse modulation method variations are explored in [41]. The major benefit of pulse modulation schemes is the capacity of achieving the needed level of dimming without using color shifts on the ejected light [42]. Nevertheless, such schemes are generally characterized by low rates of data.

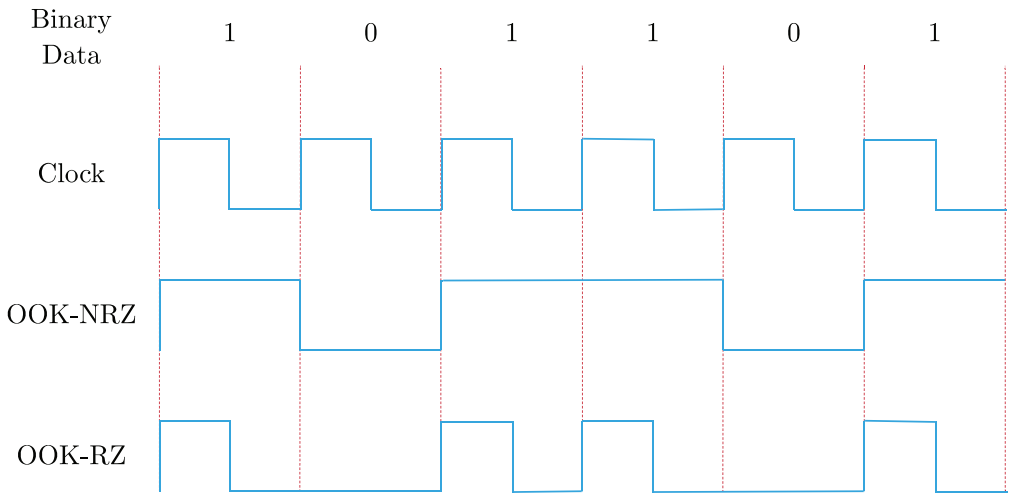

Figure 1.8: Representation of the data string “101101” using OOK-NRZ and OOK-RZ modulation schemes.

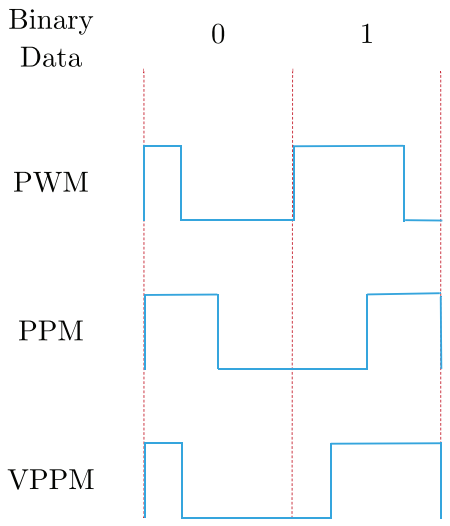

Figure 1.9: A schematic diagram showing the difference between PWM, PPM, and VPPM modulation schemes.

1.3.3 Orthogonal Frequency Division Multiplexing (OFDM)

Past modulation schemes are categorized as single-carrier modulation schemes. Single carriers are marred by high Inter-symbol interference (ISI) wherein a single symbol interrupts subsequent symbols because of nonlinearity within the VLC channels frequency response [41]. OFDM constitutes a frequency-division multiplexing (FDM) scheme utilized as the digital multicarrier modulation technique. The transmitter utilizes modulation methods that include QAM or QPSK to expand OFDM symbol's data capacity. For instance, a 16-QAM modulation maps 4 data bits to complex symbols. The complex symbols are transported by one OFDM symbol subcarrier. Afterward, several subcarriers are conveyed with each transporting various data symbols to expand the rate of data. Fig.1.10 illustrates this process. The three major challenges to OFDM within the VLC system include its restricted dimming support, the higher peak-to-average power ratio (PAPR) and the LED's nonlinearity, that is, the correlation between the LED emitted light and current is not linear. Despite such challenges, OFDM has a huge potential within VLC communication because of its immunity to inter-symbol interference and a high rate of data [41]. The basic OFDM block diagram is illustrated within Fig.1.11 wherein fast Fourier transform (FFT) is utilized at the receivers and inverse fast Fourier transform (IFFT) is utilized at OFDM transmitters.

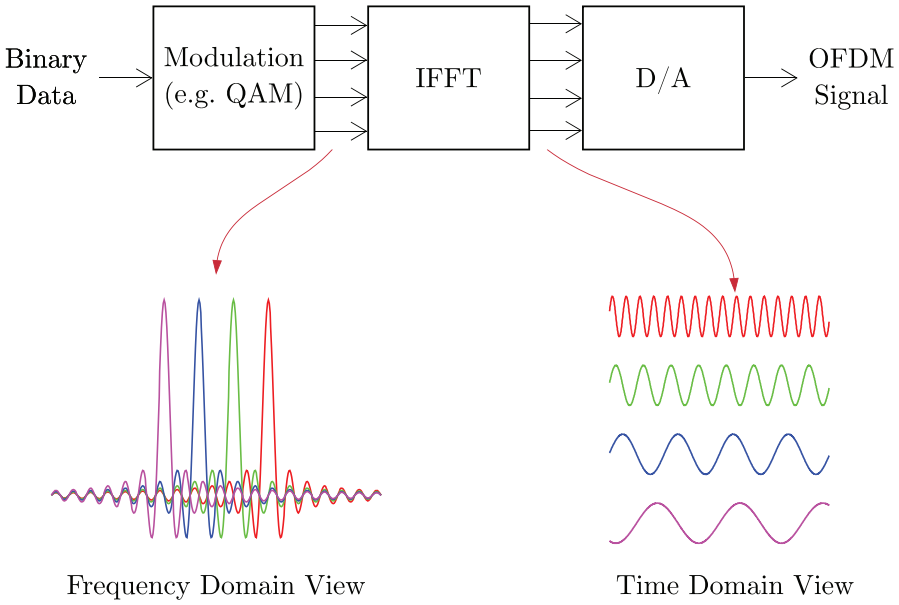

Figure 1.10: Illustration of the OFDM transmission.

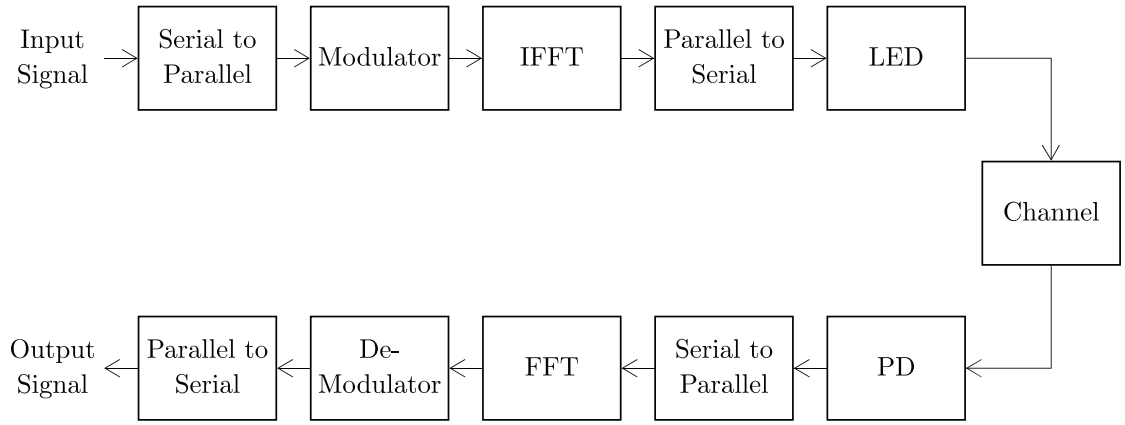

Figure 1.11: Basic OFDM System.

1.3.4 Color Shift Keying (CSK)

To address challenges emanating from other modulation schemes (for instance, restricted dimming support and low rate of data, IEEE 802.15.7 standard recommended CSK modulation which is developed mainly for VLC [43]. The block diagram for a basic CSK system is illustrated in Fig.1.12. The CSK scheme modulates the signal through the three color (blue, green, and red) intensity that tri-color LEDs (TLED) generated. The premise of the modulation is on the color space chromaticity diagram that CIE 1931 described and which is illustrated within Fig.1.13. Within CSK modulation, the bit patterns undergo encoding to wavelength (color) combinations. For instance, within the 8-CSK (refer to Fig.1.14), the source of light undergoes wavelength keying such that one out of 8 potential colors (wavelengths) is transmitted for each combined bit pair.

1.4 Multiple Access

Multiple access (MA) is utilized within VLC to grant simultaneous access to network services or resources for multiple users. Multiple accesses could be attained through three potential topologies presented within Fig.1.15 namely cellular topology with several APs shared by multiple users, single-cell topology with one AP for each room, and single –cell topology with one AP for each user [44]. An ideal topology is selected on the basis of network services, users' mobility, user number, and room size. The initial topology is feasible for users in buses, trains, or airplanes where there is one lamp (AP) on top of every seat. One cell per-room topology could be utilized in moderating

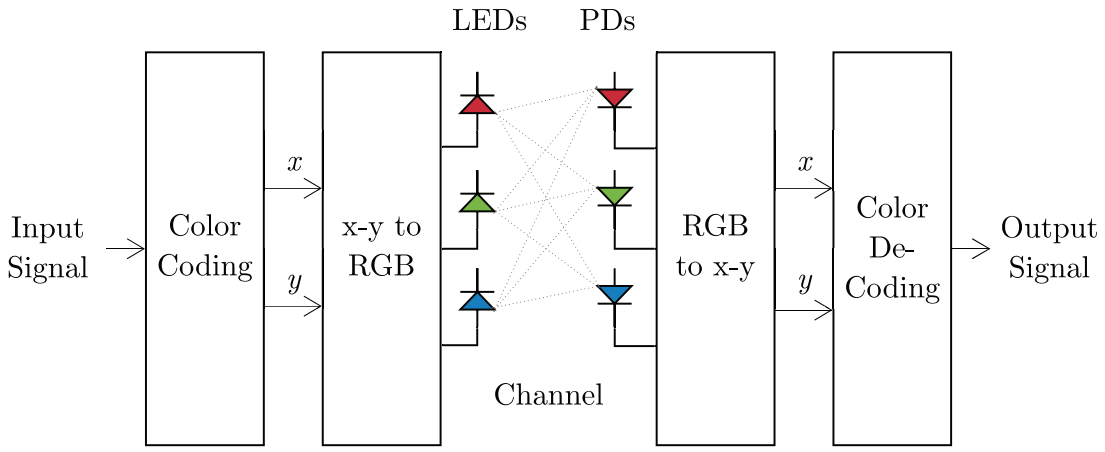

Figure 1.12: Basic TLED CSK System.

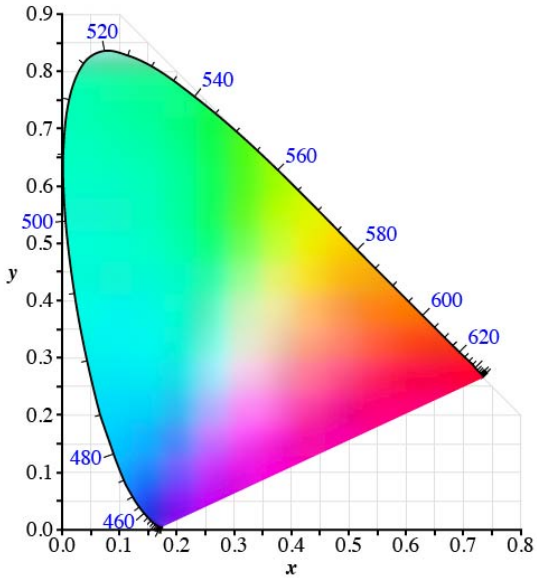

Figure 1.13: CIE 1931 color space chromaticity diagram.

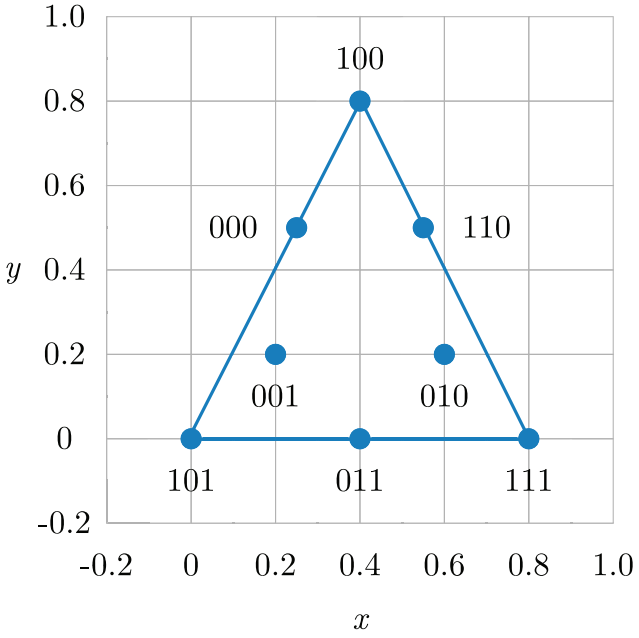

Figure 1.14: Symbol mapping for 8-CSK.

the size of offices or rooms. Cellular topologies are needed to cover airport terminals or big halls.

Numerous MA methods including non-orthogonal multiple access (NOMA), orthogonal frequency domain multiple access (OFDMA), space division multiple access (SDMA), wavelength division multiple access (WDMA), code division multiple access (CDMA), frequency division multiple access (FDMA), and time division multiple access (TDMA), have been suggested for VLC [44].

In this dissertation, we propose a new resource allocation model based on a well-known video game called the Tetris game. We have used metaheuristic algorithm such as simulated annealing (SA) simulated evolution (SimE) to optimize the RA with the objective of maximizing the system data rate under the considered partial fairness. Our Tetris model has three main contributions:

1. Reducing the time complexity of computing the cost function from $\mathcal{O}(T \times N \times M)$ to $\mathcal{O}(T)$ which has a significant impact on the running time of the optimization algorithms.

2. Improving the convergence of the optimization algorithms by considering fixed block assignments (Tetris pieces in our model) of users with the optimal data rate instead of individual slot assignments.

3. Handling the fairness objective independently from the allocation process by increasing the number of time slots assigned to users with low data rates when creating block assignments (Tetris pieces).

The rest of this dissertation is organized as follows. Chapter 2 reviews literature

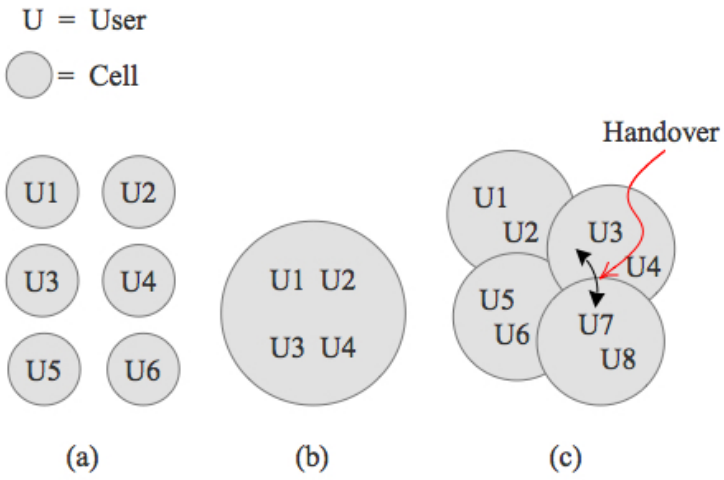

Figure 1.15: Multiple access topologies for VLC: (a) single AP per-user; (b) single AP per-room; and (c) cellular topology using multiple APs.

in the area of VLC and presents research objectives. Chapter 3 presents our Tetris-Game model to solve the resource allocation problem. The implementation details of the proposed optimization heuristics are discussed in Chapter 4. Chapter 5 discusses the experimental results of the proposed optimization heuristics.

CHAPTER 2

PROBLEM DEFINITION AND RESEARCH OBJECTIVES

In this chapter, we formulate the VLC resource allocation problem and present the research objectives. We start by a literature review of active research areas in the field of VLC systems. Then, we introduce the considered resource allocation problem and give its formal description as an optimization problem. After that, we discuss the motivation behind the selected problem. Finally, we present our research objectives.

2.1 Literature Review

As the revolutionary remedy to wireless communications containing different potential areas of application, the VLC technology has attracted the attention of investigators on a variety of levels. While studies on the physical layers are crucial in facilitating the technology, there is a need for extensive efforts within the top layers to change VLC into fully-functioning, scalable, and multi-user wireless networking technology. From

this perspective, several studies on resource allocation, load balancing, and handover were cited within the literature.

2.1.1 Handover

Handover schemes are utilized for transferring the management of a wireless transmission session that is underway from a single AP towards another. In most cases, Handover is needed within two cases [4]. Firstly, a mobile terminal exits one AP coverage area and enters another AP coverage area. Secondly, the channel of transmission is reduced because of interference or overloading. Classification of handover could be undertaken on the basis of the number of networks featured in two forms namely, *vertical handover* and *horizontal handover* [45]. Horizontal handover occurs between APs in one network. Vertical handover occurs between different network APs. Horizontal handover could be divided further into two forms based on the simultaneous connections' quantity: *soft handover* and *hard handover* [45]. In hard handover, mobile terminals could be served solely by a single AP at a given duration. Therefore, it would be delinked from the present AP prior to its connection to the following AP triggering a connectivity disruption. The key benefit of this kind is the lower complexity of the hardware and simpler implementation. Within soft handover, the mobile terminal could be served using multiple AP simultaneously. Therefore, the connection of the terminal to the existing AP is not terminated until the connection to the following AP is successful. This type provides ideal user experiences; however, it necessitates sophisticated implementation and more resources.

A number of handover schemes for VLC have been proposed over the past few years. The techniques in [46], [47] extend the traditional RSS-based handover procedures for mobile VLC systems. The work in [48], proposes a hybrid VLC and OFDMA network model in which the VLC channel is used for downlink transmission, while OFDMA channels are only used for uplinks and downlinks if there is no coverage of VLC hotspots. The proposed system combines horizontal and vertical mobile terminal handover methods to manage the mobility of users between different hotspots and OFDMA systems. In [49], soft handovers based on power and frequency are proposed to reduce data rate variations as the mobile terminal moves from cell to the other. In [50], a handover scheme combined with the joint transmission is proposed for VLC networks. The proposed scheme is compared against the data rates and the SINR of a hard-handover scheme.

2.1.2 Load Balancing

The objective of load balancing is to reduce network congestion over an area by distributing client sessions across APs with overlapping coverage. In [51], a cooperative load balancing scheme is proposed to achieve proportional fairness (PF) using distributed and centralized methods. However, the proposed scheme neglects users' mobility and handover overheads. In [52], a mobility-aware technique for load balancing is proposed to maximize system throughput, users' trajectory, and load fairness index. In [53], a dynamic load balancing scheme is proposed to reduce handovers by assigning stationary users to VLC APs and moving users to RF APs. In [54], evolutionary game theory-based load-balancing algorithm is proposed for combined VLC/RF networks

with shadowing and blocking. The work in [55] suggests a combination of load balancing and power allocation schemes for hybrid RF and VLC networks. The scheme distributes users on APs and distributes the powers of the APs on their assigned users using an iterative algorithm. The experimental results show an improved system capacity fairness with fast convergence.

2.1.3 Resource Allocation

VLC network resources such as time, power, spectrum and data rates are limited and distributed among multiple users. This limitation creates a number of problems in the allocation of resources that are interesting from both application and theoretical levels.

The work in [56] investigates a decentralized interference management scheme for OFDMA VLC system deployed inside an aircraft. In this scheme, each user equipment (UE) must transmit a busy burst (BB) in a time-multiplexed slot after receiving data successfully in order to reserve the time-frequency slot for the following frame. The access point (AP) planning to send data on a given slot must listen to its corresponding BB to decide whether to transmit or delay the transmission to another slot without any centralized controller in order to limit the interference of the co-channel. In [57], a centralized resource allocation scheme is proposed to assign the visible light multi-color logical channels to different users to minimize the interference between co-channel. A resource allocation system for VLC systems based on locations is introduced in [58]. The scheme uses a proportional fair (PF) scheduling algorithm to accommodate various transmission scenarios. In [59], a scheduling scheme is proposed

to assign LED lamps to multiple users for indoor VLC systems with the aim of minimizing inter-user interference. The authors presented a weighted graph to model the interference between users and solved the problem using a deterministic algorithm. In [60], the problem of resource allocation (RA) of moving terminals is studied on the basis of various transmission approaches to achieve relative fairness under delay conditions. The RA problem is then constructed as a non - linear programming problem (NLP) and solved through convex optimization methods. The centralized scheme proposed in [61] covers both the load balancing and the allocation of resources for the OFDM VLC system. The scheme uses a fuzzy logic algorithm to allocate resources and select an appropriate AP. A beam allocation scheme for the elimination of co-channel interference in VLC networks is proposed in [62]. In [63], a dynamic resource allocation scheme for hybrid VLC – radio frequency (RF) networks is proposed. The problem of resource optimization is implemented using the Lyapunov optimization technique. In [64], the allocation of downlink channel resources in VLC networks is studied. In [65], a resource allocation algorithm for down-link VLC networks is proposed. The problem is represented as a mixed-integer binary problem, and is solved by a centralized coordinator using Cuckoo search algorithm. An interference-free allocation system for LEDs is investigated in [66]. The work in [50] proposed a mobility management and resource allocation algorithm for VLC networks. The problem is formulated as a nonlinear integer programming problem, and is solved by means of a particle swarm optimization (PSO) algorithm.

2.2 Dynamic LED Allocation Problem

In this section, we present the dynamic LED allocation problem in visible light communication systems for both indoor and underwater scenarios. For both scenarios, we will assume a centralized light access network (C-LiAN), where all light access points are controlled by a central control unit. We will describe the setup of the considered systems in indoor and underwater environments.

2.2.1 LED Allocation for Indoor C-LiAN

In this dissertation, we consider a centralized light access network (C-LiAN) [67] with N LEDs, as illustrated in Fig. 2.1, and assume a multi-user environment with M users. A centralized unit manages the LEDs and handles the the allocation of the LED resources to users.

The transmitted signal $s_n(t)$ from the n^{th} LED is carried on a background DC light intensity with average power P_{avg} which is commonly used for light illumination. The signal $s_n(t)$ has zero mean and a variance of P_n which is the power of the LED as illustrated in Fig. 2.2 [68]. The standard deviation of the signal $s_n(t)$ is $\sqrt{P_n}$.

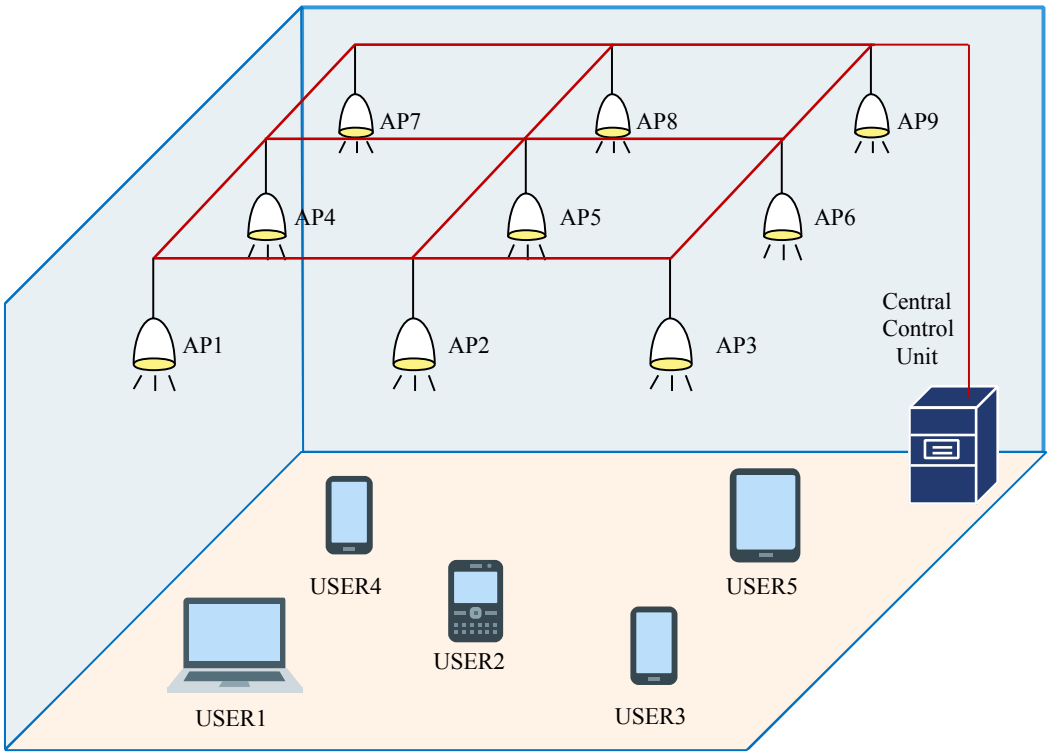

Figure 2.1: Centralized Light Access Network (C-LiAN).

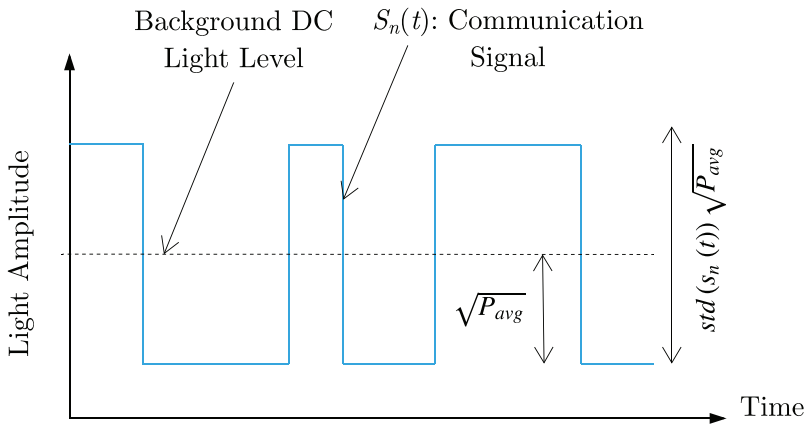

Figure 2.2: Conceptual illustration for the background DC level and communication signal on the n^{th} LED.

The received signal at the u^{th} user can be written as:

$$y_u = R\sum_{n=1}^{N}\sqrt{P_n}\alpha_{n,u}H_{n,u}x_u + R\sum_{n=1}^{N}\sum_{\substack{m=1\\m\neq u}}^{M}\sqrt{P_n}\alpha_{n,m}H_{n,u}x_m + w_u \tag{2.1}$$

where x_u is the transmitted signal to the u^{th} user, R denotes the optical to electrical conversion coefficient, P_n is the transmission power of the n^{th} LED, $H_{n,u}$ is the channel response from the n^{th} LED to the u^{th} user, and w_u denotes the Additive White Gaussian Noise ($AWGN$) with zero-mean and σ^2 variance at the u^{th} user side. The connectivity variable $\alpha_{n,u}$ is equal to 1 when the n^{th} LED is assigned to u^{th} user, and is equal to 0 otherwise. The first term denotes transmitted signal from LEDs assigned to the u^{th} and the second term is the interference from the other LEDs.

Considering the line-of-site (LOS) channel model, the channel gain between the i^{th} AP and the u^{th} user is given by [10]:

$$H_{i,u} = \begin{cases} \frac{(m+1)A}{2\pi d_{i,u}^2}\cos^m(\phi_{i,u})\cos(\psi_{i,u}) & 0 \leq \psi_{i,u} \leq \Psi_{1/2} \\ 0 & \psi_{i,u} > \Psi_{1/2} \end{cases} \tag{2.2}$$

where A denotes the photo-detector physical area, $d_{i,u}$ denotes the distance between the receiver and the transmitter, $\phi_{i,u}$ denotes the irradiance angle with regard to the transmitter perpendicular axis, $\psi_{i,u}$ denotes the incidence angle with regard to the perpendicular axis of the receiver, whereas $\Psi_{1/2}$ denotes the FOV semi-angle concentrator. The Lambertian order m is expressed as: $m = -1/\log_2(\cos(\Phi_{1/2}))$, where $\Phi_{1/2}$ is the half-power angle of the LED. For example, $\Phi_{1/2} = 60^\circ$ corresponds

to $m = 1$.

The Signal to interference and noise ratio (SINR) for the u^{th} user can be calculated as follows:

$$\mathrm{SINR}_u = \frac{E\left[\sum_{n=1}^{N} R\sqrt{P_n}\alpha_{n,u}H_{n,u}x_u\right]^2}{E\left[\sum_{n=1}^{N}\sum_{\substack{m=1\\m\neq u}}^{M} R\sqrt{P_n}\alpha_{n,m}H_{n,u}x_m\right]^2 + \sigma^2} \tag{2.3}$$

where $E[.]$ is the statistical average operator. The SINR can be approximated as in [68]:

$$\mathrm{SINR}_u = \frac{R^2\left|\sum_{n=1}^{N}\sqrt{P_n}\alpha_{n,u}H_{n,u}\right|^2}{R^2\left|\sum_{n=1}^{N}\sum_{\substack{m=1\\m\neq u}}^{M}\sqrt{P_n}\alpha_{n,m}H_{n,u}\right|^2 + \sigma^2} \tag{2.4}$$

The maximum data rate for a given user can be calculated using Shannon's channel capacity formula for noisy channels:

$$C = B\log_2(1 + \mathrm{SINR}) \tag{2.5}$$

where C is the channel capacity (maximum data rate) in bits per second and B is the channel bandwidth in hertz.

2.2.2 Sensor Allocation for Underwater C-LiAN

In this scenario, we consider a multi-user underwater C-LiAN environment with N sensors and M users (divers) as illustrated in Fig. 2.3. This scenario is similar to the indoor problem, but we will consider a different channel model with path loss and

fading.

Considering the line-of-site (LOS) channel model, the channel gain between the i^{th} sensor and the u^{th} user is given by:

$$H_{i,u} = \begin{cases} \frac{(m+1)A}{2\pi d_{i,u}^2} \cos^m(\phi_{i,u}) \cos(\psi_{i,u}) e^{-c(\lambda)d_{i,u}} & 0 \leq \psi_{i,u} \leq \Psi_{1/2} \\ 0 & \psi_{i,u} > \Psi_{1/2} \end{cases} \tag{2.6}$$

where $c(\lambda) = a(\lambda) + b(\lambda)$ is the overall attenuation, while a and b represent the scattering and absorption coefficients respectively.

Due to the complexity of the UVLC channel model, the problem may be broken down into simpler sub-problems.

Underwater Adaptive SISO VLC: In this sub-problem, we study bit loading and power loading for a single-input-single-output (SISO) underwater link. In this setup, we consider a single transmitter and a single receiver in the presence of turbulence fading. The objective function of this problem is to minimize the power budget subject to data rate constraints.

Underwater Adaptive MIMO VLC: In this sub-problem, we study the same problem for multiple-inputs multiple-outputs (MIMO) underwater links in the presence of turbulence fading. We consider the same objective function that is the minimization of power budget subject to data rate constraints.

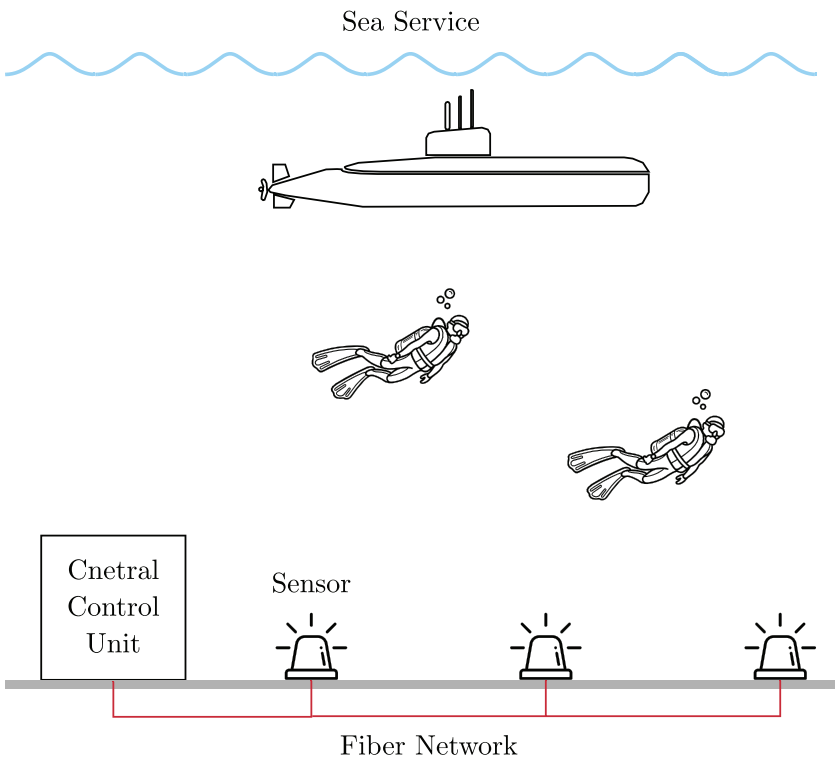

Figure 2.3: A multi-user underwater C-LiAN environment.

2.3 Problem Definition

As indicated in the previous section, we consider a centralized light access network (C-LiAN) environment with N LEDs and M users. Each LED uses the same spectrum and time-division multiple access (TDMA) is assumed in downlink transmission. One transmission frame includes T time slots as illustrated in Fig. 2.4. Hence, the maximum allowed serving user count is equal to $T \times N$ because one user is allocated only one LED and only one-time slot. However, based on the total number of users, multiple LEDs can be allocated to a single user within the same time slot, which is also called "Coordinated Multi-point Transmission."

Next, we formulate the allocation problem as an optimization problem by defining the three optimization elements: decision variables, constraints, and objective functions.

Decision Variables: We define a time-varying allocation matrix, $A[t]$, as our decision variable. Each element $\alpha_{n,u}[t]$ of $A[t]$ takes a value of 0 or 1, where $\alpha_{n,u}[t] = 1$ indicates that the n^{th} LED is assigned to the u^{th} user in the t^{th} time slot.

Constrains: In this problem, we consider two constraints:

1. Since the time-slot of an LED cannot be shared with multiple users, the following equation should be satisfied while optimizing the allocation matrix:

$$\sum_{u=1}^{M} \alpha_{n,u}[t] \leq 1; 1 \leq n \leq N, 1 \leq t \leq T \tag{2.7}$$

	SLOT#1	SLOT#2	SLOT#3	. . .	SLOT#T-1	SLOT#T
LED#1	UE#2	UE#6	UE#8	. . .	UE#4	UE#7
LED#2	UE#3	UE#2	UE#2	. . .	UE#8	UE#1
LED#3	UE#2	UE#7	UE#1	. . .	UE#4	UE#7
. . .	. . .	. . .	. . .	. . .	. . .	. . .
LED#N-1	UE#2	UE#2	UE#9	. . .	UE#1	UE#9
LED#N	UE#5	UE#1	UE#9	. . .	UE#2	UE#6

Figure 2.4: A transmission frame with T time slots.

2. In order to serve a user by at least one LED within at least one time-slot, the following equation should be satisfied:

$$\sum_{n=1}^{N}\sum_{t=1}^{T}\alpha_{n,u}[t] \geq 1; 1 \leq u \leq M \tag{2.8}$$

The above constraints are valid under the assumption that the maximum $T \times N$ users exist in the indoor room environment under consideration.

Objective Functions: In this problem, we consider the objective function of maximizing the system data rate under the consideration of partial fairness using the following equation:

$$\max_{A[t]} \left(\sum_{u=1}^{M}\sum_{t=1}^{T} \log_2 \left(1 + \mathrm{SINR}_u[t]\right) \right) \tag{2.9}$$

2.4 Motivation

Visible light communication (VLC) is a growing technology that can overcome the limitations of the RF spectrum for wireless communication systems. VLC systems have many advantages over RF-based communication systems such as broad-spectrum, safety, low cost, energy efficiency, and information security. As a revolutionary solution to wireless communications with a wide range of applications, the VLC technology has attracted researchers on various levels such as the physical layer, handover, load balancing, and resource allocation. While research on the physical layer is essential in enabling this technology, more efforts are needed in the upper layer to transform VLC into a multi-user, scalable, and fully functioning wireless networking technology. In

this context, we have chosen to contribute to this evolving research area concentrating on the optimization problems of resource allocation.

2.5 Research Objectives

The allocation of resources in VLC systems is an attractive research area. Most of the proposed schemes use deterministic algorithms to solve NP-hard allocation problems. Our primary research objective is to explore the application of meta-heuristics for optimizing a load-aware dynamic LED allocation system in C-LiAN. The potential optimization problem to be pursued during the research is the maximization of system data rate under the considered partial fairness. In addition to that, we will study a similar problem to assign sensors to divers in underwater C-LiAN dynamically.

CHAPTER 3

THE TETRIS GAME MODEL

In this dissertation, we consider a resource allocation problem (RA) for an indoor room environment with N LEDs and a multi-user environment with M users. A centralized unit manages the LEDs and handles allocation of the LED resources to users. Each LED uses the same spectrum and TDMA is assumed in downlink transmission. One transmission frame includes T time-slots. Hence, maximum allowed serving user count is equal to $T \times N$ considering the fact that one user is allocated with only one LED and only one-time slot. However, based on the total number of users, multiple LEDs can be allocated to a single user within the same time slot, which is also called as "Coordinated Multipoint Transmission". We have modeled this problem as a Tetris game and used simulated annealing (SA) to optimize the RA with the objective of maximizing system data rate under the considered partial fairness. Our Tetris model has the advantage of improving the convergence of the optimization algorithms by considering various block assignments (Tetris pieces in our model) instead of individual slot assignments.

3.1 The Tetris Game

Tetris is a video game developed by Alexey Pajitnov in 1984. The game is played on a board composed of R rows and C columns, in which pieces of 7 different shapes drop from top to bottom as shown in Fig. 3.1. The optimization of the offline version of the Tetris game is proven to be NP-complete [69]. The optimization objectives of this problem include: maximizing the number of canceled rows, maximizing the number of packed pieces, and minimizing the number of rows occupied by pieces.

Example 1. In this example, we illustrate the mapping between the RA problem and the Tetris problem. Consider a room with four LEDs ($N = 4$) and four users ($M = 4$); and assume that the channel gains are given by Table 3.1. Further, assume that each transmission frame includes four time-slots ($T = 4$). A total of 16 time-slots will be equally divided between the four users (4 time-slots per user). Table 3.2 shows all possible LED allocations for each user with allocated LEDs hightailed in color. The assignments are sorted in descending order of their data rates which are calculated using equation (2.5). In our Tetris model, each user's assignment represents a piece consisting of LEDs assigned to that user with a weight corresponding to its data rate. Finding the assignment that maximizes the overall data rate is equivalent to finding an optimal packing of the pieces in a fixed height Tetris board such that the total weight is maximized. Note that in order to satisfy the constraint in equation (2.8), we need to pack at least one piece for each user in the Tetris board. Fig 3.2 shows an optimal packing of the Tetris pieces from Table 3.2 in four rows (i.e., four time-slots). The packing includes one piece with 4 LEDs for each one of the 1^{st} and 3^{rd} users, and

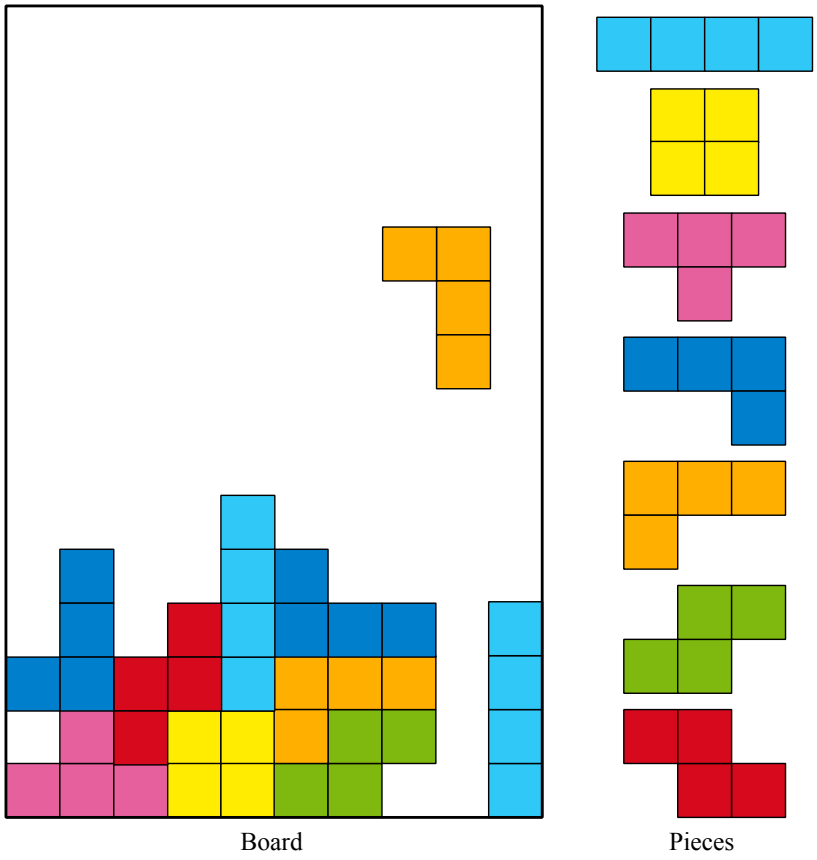

Figure 3.1: An example of Tetris board with seven pieces (shapes).

two pieces with 2 LEDs for each one of the 2^{nd} and 4^{th} users.

3.2 The Tetris Model

Given an instance of the RA problem with N LEDs, M users, T time-slots, one can map this problem into a Tetris game with p pieces and an $m \times n$ board, where $n = N$ and $T \leq m \leq p$. The model uses two structures: one to store the pieces (allocation combinations) and the other to store the pieces packed in the Tetris board (allocation solution).

3.2.1 Tetris Pieces

The Tetris pieces can be represented as an array $Tetris$ of p elements (structures). Each element $Tetris[k]$ represents one piece in the Tetris game model which contains the following members:

- Piece (p_k): an $n-$tuple of the form $(a_1, a_2, \ldots, a_n)$ which represents an allocation to a user u_k, where $a_i = 1$ if the i^{th} LED is allocated for the user and 0 otherwise.
- User (u_k): the user associated with the piece p_k.

Fig. 3.3 shows the array $Tetris$ representing the pieces in Table 3.2.

3.2.2 Solution Representation

In a direct implementation of the RA problem, a solution S can be represented as $N \times T$ matrix where the element $S[i, j] = u$ indicates that the i^{th} LED is assigned to

Table 3.1: Channel gains ($\times 10^{-5}$) for users in Example 1.

	D1	**D2**	**D3**	**D4**
UE1	0.0966	0.2943	0.0884	0.2527
UE2	0.4251	0.0885	0.1876	0.0584
UE3	0.1703	0.0528	0.4608	0.0863
UE4	0	0	0.1997	0.176

Table 3.2: All possible combinations of Tetris pieces for users in Example 1.

UE	#	D1	D2	D3	D4	D. Rate
1	1	1	1	1	1	9.716115
	2	1	1	0	1	5.644801
	3	0	1	1	1	5.375191
	4	0	1	0	1	3.260951
	5	1	1	1	0	2.189977
	6	1	0	1	1	1.675555
	7	1	1	0	0	1.205903
	8	0	1	1	0	1.133929
	9	1	0	0	1	0.871187
	10	0	0	1	1	0.813839
	11	0	1	0	0	0.536918
	12	0	0	0	1	0.352892
	13	1	0	1	0	0.155868
	14	1	0	0	0	0.032908
	15	0	0	1	0	0.026907

UE	#	D1	D2	D3	D4	D. Rate
3	31	1	1	1	1	9.862782
	32	1	0	1	1	7.242535
	33	1	1	1	0	5.877301
	34	1	0	1	0	4.387159
	35	0	1	1	1	3.716833
	36	0	0	1	1	2.795596
	37	0	1	1	0	2.312496
	38	0	0	1	0	1.679941
	39	1	1	0	1	0.535393
	40	1	0	0	1	0.320812
	41	1	1	0	0	0.221292
	42	1	0	0	0	0.111565
	43	0	1	0	1	0.068339
	44	0	0	0	1	0.022788
	45	0	1	0	0	0.007772

UE	#	D1	D2	D3	D4	D. Rate
2	16	1	1	1	1	9.823388
	17	1	1	1	0	6.934416
	18	1	0	1	1	5.759876
	19	1	0	1	0	4.161001
	20	1	1	0	1	3.340802
	21	1	1	0	0	2.40917
	22	1	0	0	1	2.014946
	23	1	0	0	0	1.381555
	24	0	1	1	1	0.69346
	25	0	1	1	0	0.406161
	26	0	0	1	1	0.297434
	27	0	0	1	0	0.147113
	28	0	1	0	1	0.08054
	29	0	1	0	0	0.024832
	30	0	0	0	1	0.009974

UE	#	D1	D2	D3	D4	D. Rate
4	46	0	0	1	1	7.796453
	47	0	0	1	0	1.177687
	48	0	0	0	1	0.818933

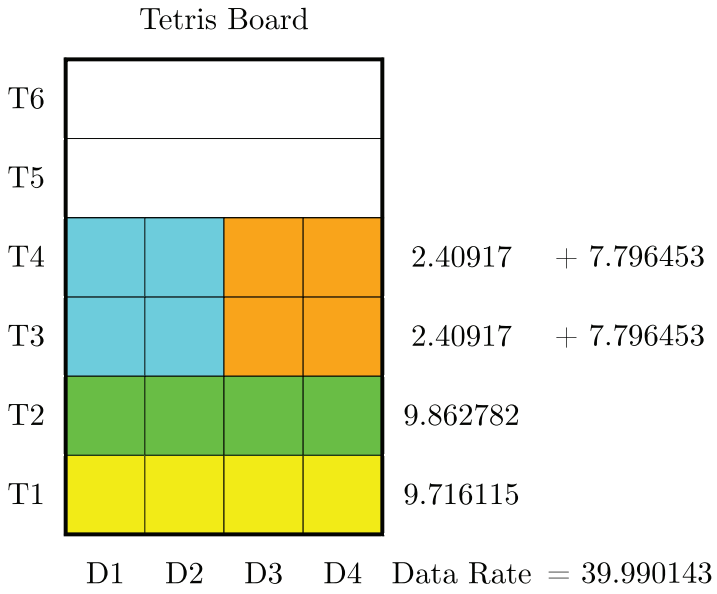

Figure 3.2: An example of modeling RA as a Tetris game.

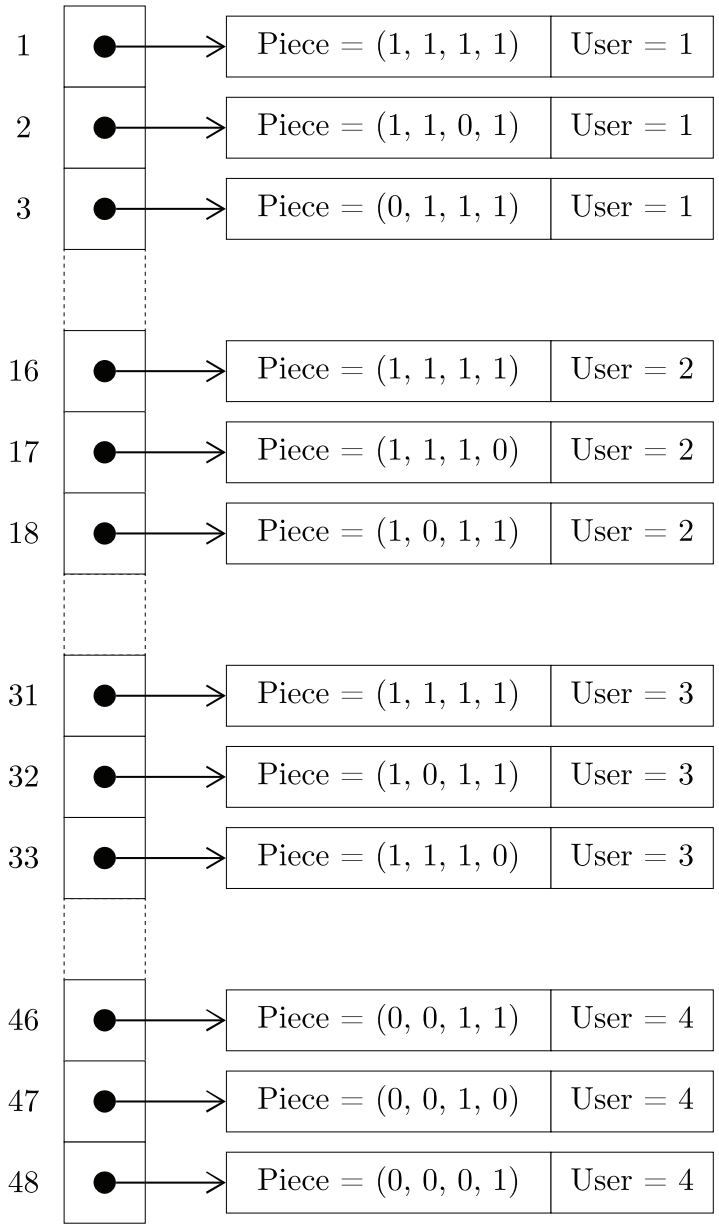

Figure 3.3: Representation of the pieces in Table 3.2.

the u^{th} user in the j^{th} time slot. Table 3.3 shows the representation of the solution in Fig. 3.4.

A solution of the RA problem modeled as a Tetris game can be represented as an array S of m elements (structures), where $T \leq m \leq p$. The element $S[r]$ represents the assignment to the r^{th} row of the Tetris board and it contains the following members:

- Assignment (A_r): an $n-$tuple of the form (a_1, a_2, ..., a_n) combining (ORing) the pieces placed in to the r^{th} row.
- Pieces (P_r): a list of pieces placed in to the r^{th} row.

Fig. 3.4 illustrates the solution representation of the assignment in Example 1 (see Fig 3.2). Each solution S is associated with a weight (W_s) proportional to the total data rate of the pieces placed in to rows (time-slots) from 1 to T. This weight is calculated as follows:

$$C_s = B \sum_{u=1}^{M} \sum_{t=1}^{T} \log_2 \left(1 + \mathrm{SINR}_u[t]\right) \tag{3.1}$$

$$W_s = C_s \frac{M^{'}}{M} \tag{3.2}$$

where C_s is the total data rate of S, M is the total number of users, and $M^{'}$ is the total number of users with allocated time-slots between 1 and T. Note that the ratio $M^{'}/M$ is used to penalize solutions violating the constraint in equation (2.8).

Table 3.3: A representation of the solution in Example 1 using a direct implementation.

	T#1	T#2	T#3	T#4
LED#1	1	3	2	2
LED#2	1	3	2	2
LED#3	1	3	4	4
LED#4	1	3	4	4

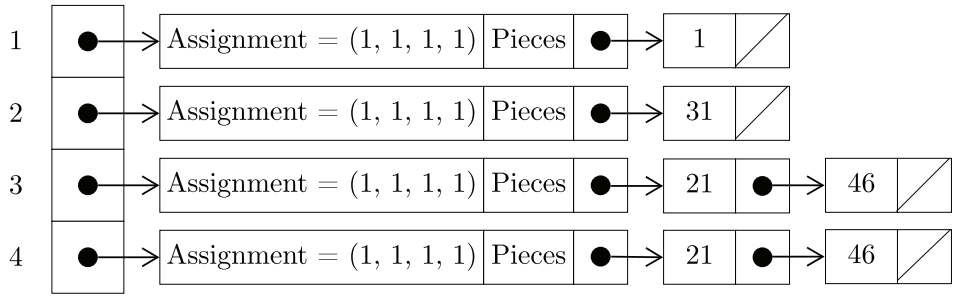

Figure 3.4: A solution representation of the assignment in Example 1.

3.2.3 Extended Solution Representation

As discussed in section 3.2.2, we have two representations of the RA solution: one for the direct implementation and one for the Tetris-Game model. In the direct implementation, there exists a total of $N \times T$ slots that can be allocated to M users. If the slots are equally divided between users, then each user will have $(N \times T)/M$ slots. Since the objective is to maximize the system data rate under the consideration of partial fairness, some users may be allocated more slots than others as long as the constraint in equation (2.8) is respected. Therefore, we decided to use $K \times T$ time-slots instead of T time-slots as defined in the original problem, where K is an integer greater than 1. This allows the initial solution generator to include extra assignments for each user $(K \times N \times T)/M$. Then, it is up to the optimization algorithm to allocate the slots to maximize the total data rate in the first T time-slots. Note that the assignments in the remaining $(K \times T - T)$ time-slots are not included in the calculation of the total data rate, hence, they should be occupied by assignments with the least data rates. The extended solution representation for direct implementation of the RA problem is illustrated by an example in Fig. 3.5.

In the Tetris-Game model, the extension of the solution is done in a different manner. Instead of using extra time-slots, we repeat each generated piece K times, where K is an integer that can be selected based on experimental results. Then, the optimization algorithm will try to pack the pieces into T rows of the Tetris board such that the total data rate is maximized. The remaining pieces placed into rows grater than T will not participate in the calculation of the total data rate.

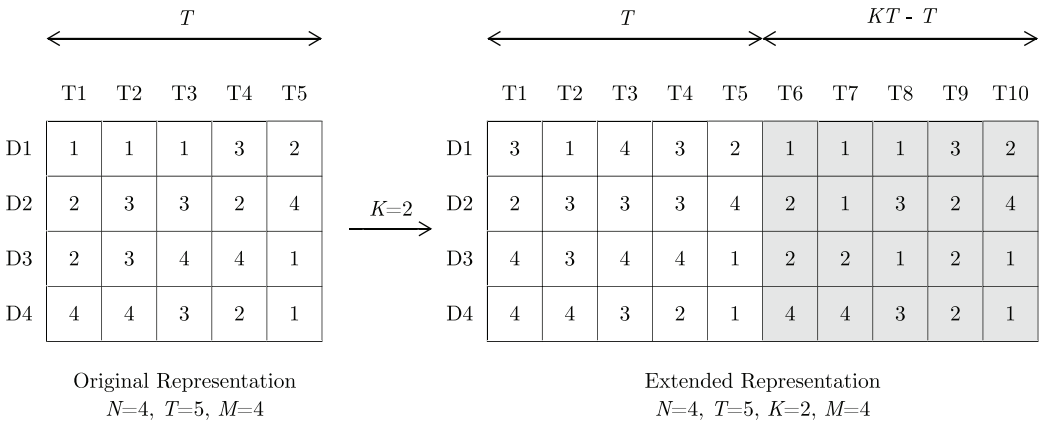

Figure 3.5: An example of the proposed extended solution representation.

3.3 Generating Tetris Pieces

Our experiments show that metaheuristic algorithms converge faster when using the Tetris model allocation as compared to the individual LED allocation. However, generating all possible combinations of the pieces is intractable for practical problems. This makes individual allocation more flexible in exploring the various assignment combinations. In order to take advantage of both approaches, we use large pieces with high weights and small pieces (single LED assignments) with low weights. The large pieces speed up the search process, while small pieces enable the optimization algorithm to explore more areas by merging individual assignments to form pieces with new combinations. In Example 1, we can generate the assignment of the 2^{nd} user in $T3$ using either a single piece with $(1, 1, 0, 0)$ or using two pieces with $(1, 0, 0, 0)$ and $(0, 1, 0, 0)$ respectively.

When generating the pieces for each user, we consider only LEDs with nonzero channel-gain. In our Tetris model, we generate two sets of pieces for each user:

1. *Single LED Allocation*: consists of all combinations of pieces with a single LED allocation. In Example 1, the first three users have four possible pieces $\{(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)\}$, while the last user has only two possible pieces $\{(0, 0, 1, 0), (0, 0, 0, 1)\}$.

2. *Multiple LEDs Allocation*: consists of pieces with group of $(2, 3, \ldots, N)$ LEDs selected based on their channel-gains ordered from highest to lowest. In Example 1, the first user has the following channel-gains (0.0966, 0.2943, 0.0884, 0.2527), and his set of group allocations is given by $\{(0, 1, 0, 1), (1, 1, 0, 1), (1, 1, 1, 1)\}$.

Algorithm 1 shows how to generate the two sets of pieces for each user.

3.4 Placement of Pieces

In this section, we will discuss some terminologies for placing a piece in the Tetris board:

- A piece p is said to be overlapping with an assignment A if there exists $p^{'} \in A$ such that $(a_i^{'} = a_i = 1)$ for some i $(1 \leq i \leq N)$.
- Testing for the overlapping between a piece p and an assignment A is simply an AND operation. We say that p overlaps with A if $p \wedge A = 0$.
- Adding a piece p to an assignment A is simply an OR operation $(p \vee A)$.
- Removing a piece p from an assignment A is simply an AND operation with the complement of the piece $(\neg p \vee A)$.
- The value of the piece that can fill the empty slots in an assignment A is given by $\neg A$.

Algorithm 1: Procedure for generating the Tetris pieces

Input : M, N, H
Output: $Tetris$

$k \leftarrow 1$
for $i \leftarrow 1$ **to** M **do**
  /* Get channel-gains corresponding to the i^{th} user */
  $h \leftarrow \{H[i,1], H[i,2], \ldots, H[i,N]\}$
  /* Get the indices of nonzero channel-gains in h */
  $h_p \leftarrow \{r | h[r] > 0, \text{for } 1 \leq r \leq N\}$
  /* Get the size of h_p */
  $N_p \leftarrow |h_p|$
  /* Sort h_p in descending order of the values of h */
  $(indices, values) \leftarrow Sort(h)$
  $h_p \leftarrow indices[1..N_p]$
  /* Generate single allocation set */
  for $j \leftarrow 1$ **to** N_p **do**
    $d \leftarrow h_p[j]$
    $Tetris[k].Pieces \leftarrow (0, 0, \ldots, 0)$
    $Tetris[k].Pieces[d] \leftarrow 1$
    $Tetris[k].User \leftarrow i$
    $k \leftarrow k + 1$
  end
  /* Generate multiple allocations set */
  for $j \leftarrow 2$ **to** N_p **do**
    $Tetris[k].Pieces \leftarrow (0, 0, \ldots, 0)$
    $Tetris[k].User \leftarrow i$
    for $m \leftarrow 1$ **to** j **do**
      $d \leftarrow h_p[m]$
      $Tetris[k].Pieces[d] \leftarrow 1$
    end
    $k \leftarrow k + 1$
  end
end
return $Tetris$

CHAPTER 4

OPTIMIZATION HEURISTICS BASED ON TETRIS GAME MODEL

In order to demonstrate the effectiveness of our resource allocation model, we have developed two metaheuristics for the indoor resource allocation problem. The first algorithm is a direct application of the simulated annealing (DASA), and the second is a Tetris-Game based simulated annealing (TGSA). The implementation details of these algorithms are discussed in this chapter.

4.1 Initial Solution Generation

The simulated annealing (SA) starts from a random initial solution. In this section, we discuss the implementation of the procedures that generate the initial solution for both direct implementation and Tetris-Game model.

In the direct implementation of the RA problem, the initial solution is generated randomly as shown in Algorithm 2. The procedure starts by generating an array A from the set of elements $\{1, 2, \times M\}$ with each element repeated n times, where n is the number of slots per user. If $(K \times N \times T)$ is not multiple of M, then tabbing is used to fill the remaining elements of A. Next, the procedure rearranges the elements of A in random order and assigns them to the slots in the solution S_{init}.

Algorithm 2 shows the procedure for generating the initial solution for the Tetris-Game model. The procedure starts by generating an array A from set of elements $\{1, 2, \times n\}$ with each element repeated K times, where n is the number of pieces in the $Tetris$ array. Next, the procedure rearranges the elements of A in random order and packs them into the Tetris board (i.e., the solution) in a first-fit manner. A piece can be packed into a row if and only if it does not overlap with pieces already packed in that row.

4.2 Simulated Annealing

Simulated annealing (SA) is a metaheuristic introduced by Kirpatrick et al. [70] in 1983, and independently by Černy [71] in 1985 to solve optimization problems with large search spaces. The general structure of the SA and the Metropolis procedure are shown in Algorithms 4 and 5 respectively [72]. The input to the algorithm consists of 6 parameters (S_0, T_0, α, β, M_0, $MaxTime$). The algorithm starts from an initial solution S_0 which can be generated randomly. In the beginning of the procedure, the temperature is set to an initial value T_0, and is decreased gradually using a cooling

rate α. The value of α is typically chosen in the range of $0.8 \leq \alpha \leq 0.99$. The initial value of the time spent in the annealing process at a certain temperature is given by the parameter M_0. This time is gradually increased as the temperature decreases using the parameter β. The total time for the annealing process is represented by the parameter $MaxTime$.

The Metropolis procedure mimics the annealing operation at a certain temperature T. This procedures receives 6 inputs which are the current solution and its cost (S_{cur}, C_{cur}), the best solution seen so far and its cost (S_{best}, C_{best}), the current temperature T, and the parameter M which represents the amount of time need to be spent by the annealing process at temperature T. The Metropolis procedure uses the procedure *Neighbor* to generate a new solution S_{new} by doing a minor modification to the current solution S_{cur}. The cost of the new solution is evaluated by the function *Cost*. If the cost of the new solution S_{new} is better than the cost of the current solution S_{cur}, then the current solution is updated by accepting the new solution. The same statement applies to the best solution seen so far S_{best}. The Metropolis procedure may accept a solution with a higher cost with a probability $p < e^{-\Delta C/T}$ where ΔC is the difference between the costs of the new and current solutions.

4.3 Implementation Details

In this section, we discuss the implementation details of the main functions of the simulated annealing for both DASA and TGSA.

4.3.1 The Neighbor Function

In the SA algorithm, the neighbor function generates a new solution by making a minor modification to the current solution. In the direct application of the SA algorithm to the RA problem, we have implemented a simple neighbor function that exchanges the assignment of two time-slots selected randomly as illustrated in Algorithm 6. Such approach can not be used in the Tetris model as it may generate invalid solutions due to overlapping pieces. Thus, before placing a piece p in a row r, the neighbor function needs to make sure that p does not overlap with any piece already in r.

The neighbor function for the Tetris-Game model is described in Algorithm 7. This function selects a random piece p_{out} from a random row t_{out}, and places it into another random row t_{in}. The function checks for overlapping between p_{out} and existing pieces in t_{in}, and moves those pieces to t_{out}. The function repeats the process of exchanging overlapping pieces between t_{out} and t_{in} until there are no more overlaps. The row t_{out} is selected randomly from 1 to T, while t_{in} is selected randomly from 1 to T_s, where T_s is the total number of rows in the Tetris board (i.e., Tetris solution). In this way, the TGSA algorithm can pack the pieces into T rows (time-slots) while maximizing the total data rate.

4.3.2 The Cost Function

The cost function in SA computes the objective function to be minimized or maximized. In the RA problem, our objective is to maximize the total data rate under the consideration of partial fairness as given in equation (2.9. The objective function can

be computed is shown in Algorithm 9. We have implemented two cost functions for both DASA and TGSA:

1. *Full Cost Function*: evaluates the objective function for the whole solution based on equation (2.9) as shown in Algorithms. This function needs to be applied only once on the initial solution.

2. *Partial Cost Function*: evaluates the objective function by recomputing the SINR values for the modified slots of the neighbor solution as shown in Algorithms.

In the DASA, the cost of the new solution (C_{new}) is computed based on the on the cost of the old solution (C_{old}) using the following formula:

$$C_{new} = \begin{cases} C_{old} + \sum\limits_{\substack{u \in \{u_x, u_y\} \\ t \in \{t_x, t_y\}}} \log_2(1 + \text{SINR}_{new}[u,t]) & \\ \qquad - \sum\limits_{\substack{u \in \{u_x, u_y\} \\ t \in \{t_x, t_y\}}} \log_2(1 + \text{SINR}_{old}[u,t]) & t_x \leq T \\ C_{old} + \sum\limits_{\substack{u \in \{u_x, u_y\} \\ t = t_y}} \log_2(1 + \text{SINR}_{new}[u,t]) & \\ \qquad - \sum\limits_{\substack{u \in \{u_x, u_y\} \\ t = t_y}} \log_2(1 + \text{SINR}_{old}[u,t]) & t_x > T \end{cases} \tag{4.1}$$

where u_x and u_y are the two exchanged user assignments in time-slots t_x and t_y respectively, and SINR_{new} and SINR_{old} are the SINR values computed based on the assignments in the new and old solutions respectively. Note that assignments in time-slots greater than T do not affect the calculation of the cost function.

In the TGSA, the cos tis computed using the following formula:

$$C_{new} = \begin{cases} C_{old} + \sum\limits_{\substack{u \in \{u_{in} \cup u_{out}\} \\ t \in \{t_{in}, t_{out}\}}} \log_2(1 + \text{SINR}_{new}[u,t]) & \\ \qquad - \sum\limits_{\substack{u \in \{u_{in} \cup u_{out}\} \\ t \in \{t_{in}, t_{out}\}}} \log_2(1 + \text{SINR}_{old}[u,t]) & t_{out} \leq T \\ C_{old} + \sum\limits_{\substack{u \in \{u_{in} \cup u_{out}\} \\ t = t_{in}}} \log_2(1 + \text{SINR}_{new}[u,t]) & \\ \qquad - \sum\limits_{\substack{u \in \{u_{in} \cup u_{out}\} \\ t = t_{in}}} \log_2(1 + \text{SINR}_{old}[u,t]) & t_{out} > T \end{cases} \tag{4.2}$$

where u_{in} and u_{out} are the set of users assigned to time-slots t_{in} and t_{out} respectively.

Algorithm 2: Generates the initial solution in the direct implementation

Input : M, N, T, K
Output: S_{init}

```
1  /* Determine the number of slots available for each user        */
2  n ← (K × N × T)/M
3  /* Generate an array from the set of users with each user repeated
       n times                                                      */
4  A ← {1, 2, ..., M}^n
5  /* Arrange the elements of A in random order                    */
6  A ← RandomPermutation(A)
7  /* Assign the elements of A to the slots in the solution        */
8  k ← 1
9  for i ← 1 to N do
10 |   for j ← 1 to K × T do
11 |   |   S_init[i, j] ← A[k]
12 |   |   k ← k + 1
13 |   end
14 end
15 return S_init
```

Algorithm 3: Generates the initial solution in the Tetris-Game model

Input : M, N, T, K, $Tetris$
Output: S_{init}

$n \leftarrow length(Tetris)$
/* Generate an array from the set of piece's indices in $Tetris$ with each index repeated K times */
$A \leftarrow \{1, 2, \ldots, n\}^K$
/* Arrange the elements of A in random order */
$A \leftarrow RandomPermutation(A)$
/* Initialize the solution */
for $i \leftarrow 1$ **to** $K \times n$ **do**
    $S[i].Assignment \leftarrow (0, 0, \ldots, 0)$
    $S[i].Pieces \leftarrow \phi$
end
/* Pack the pieces idexed by A into the rows of S_{init} in a first-fit manner */
$k \leftarrow 1$
for $i \leftarrow 1$ **to** $K \times n$ **do**
    $index \leftarrow A[k]$
    $piece \leftarrow Tetris[index].Piece$
    $k \leftarrow k + 1$
    for $j \leftarrow 1$ **to** $K \times n$ **do**
        /* Check for overlapping between the piece and the assignment in the j^{th} row */
        $overlap \leftarrow S[j].Assignment \wedge piece$
        if $\neg overlap$ **then**
            /* Merge the piece with the current assignment */
            $S[j].Assignment \leftarrow S[j].Assignment \vee piece$
            /* Add the index of the piece to the list of pieces in the j^{th} row */
            $S[j].Pieces \leftarrow S[j].Pieces \cup \{index\}$
            break
    end
end
/* Copy none empty rows in S into S_{init} */
$k \leftarrow 1$
for $i \leftarrow 1$ **to** $K \times n$ **do**
    if $S[i].Assignment \neq (0, 0, \ldots, 0)$ **then**
        $S_{init}[k] = S[i]$
        $k \leftarrow k + 1$
end
return S_{init}

Algorithm 4: Simulated annealing algorithm

Input : S_0, T_0, α, β, M_0, $MaxTime$
Output: S_{best}

$T = T_0$
$M = M_0$
$S_{cur} = S_0$
$S_{best} = S_{cur}$
$C_{cur} = Cost(S_{cur})$
$C_{best} = C_{cur}$
$Time = 0$
repeat
    Call $Metropolis(S_{cur}, C_{cur}, S_{best}, C_{best}, T, M)$
    $Time = Time + M$
    $T = \alpha T$
    $M = \beta M$
until $Time \geq MaxTime$;
return S_{best}

Algorithm 5: The Metropolis procedure

Input : S_{cur}, C_{cur}, S_{best}, C_{best}, T, M
Output: S_{cur}, C_{cur}, S_{best}, C_{best}

repeat
    $S_{new} = Neighbor(S_{cur})$
    $C_{new} = Cost(C_{cur}$
    $\Delta C = C_{new} - C_{cur}$
    if $\Delta C < 0$ **then**
        $S_{cur} = S_{new}$
        if $C_{new} < C_{best}$ **then**
            $S_{best} = S_{new}$
        end
    else
        if $RANDOM < e^{-\Delta C/T}$ **then**
            $S_{cur} = S_{new}$
        end
    end
    $M = M - 1$
until $M = 0$
return S_{cur}, C_{cur}, S_{best}, C_{best}

Algorithm 6: The neighbor function for DASA Algorithm

Input : S, N, T, K
Output: S

```
/* Select two different random slots from S                    */
repeat
    t_x ← RandomInteger(1..K × T)
    t_y ← RandomInteger(1..T)
    d_x ← RandomInteger(1..N)
    d_y ← RandomInteger(1..N)
until t_x ≠ t_y or d_x ≠ d_y
/* Exchange the two slots                                       */
Swap(S[d_x, t_x], S[d_y, t_y])
return S
```

Algorithm 7: The neighbor function for TGSA Algorithm

Input : S, N, T, $Tetris$
Output: S

```
/* Get the total length of the solution S                    */
```
$N_s \leftarrow |S|$
```
/* Select two different random rows from S                   */
```
repeat
  $t_{out} \leftarrow RandomInteger(1..T_s)$
  $t_{in} \leftarrow RandomInteger(1..T)$
until $t_{out} \neq t_{in}$
```
/* Select a random piece from t_out                          */
```
$p_{out} \leftarrow S[t_{out}].Pieces[RandomInteger(1..N_s)]$
```
/* Initialize lists                                          */
```
$check\_list \leftarrow \{p_{out}\}$
$move\_list\_pieces \leftarrow \phi$
$move\_list\_rows \leftarrow \phi$
$MoveList(check\_list, move\_list\_pieces, move\_list\_rows, t_{in}, t_{out}, S, Tetris)$
```
/* Place pieces in move list in their proper rows            */
```
for each $p \in move\_list\_pieces$ and $t \in move\_list\_rows$ **do**
```
  /* Get the piece details from the Tetris                   */
```
  $piece \leftarrow Tetris[p].Piece$
```
  /* Place the piece                                         */
```
  $S[t].Assignment = piece \vee S[t].Assignment$
  $S[t].Pieces = S[t].Pieces \cup p$
end
```
/* Check if t_out row is empty, then delete it               */
```
if $S[t_{out}].Pieces = \phi$ **then**
  $Delete(S[t_{out}])$
end
return S

Algorithm 8: The MoveList procedure

Input : $check\_list$, $move\_list\_pieces$, $move\_list\_rows$, t_{in}, t_{out}, S, $Tetris$
Output: $move\_list\_pieces$, $move\_list\_rows$

```
1  overlap_list ← φ
2  /* Determine overlaps between t_in and t_out due to moving p_out      */
3  while check_list ≠ φ do
4      /* Remove one element from the check list                         */
5      p_out ← check_list[1]
6      check_list ← check_list − p_out
7      /* Get the corresponding piece from Tetris                       */
8      piece_out ← Tetris[p_out].Piece
9      /* Remove the piece from t_out                                   */
10     S[t_out].Assignment ← S[t_out].Assignment ∧ ∄piece
11     /* Add p_out to move lists in order to place it at t_in          */
12     move_list_pieces ← move_list_pieces ∪ p_out
13     move_list_rows ← move_list_rows ∪ t_in
14     /* Find pieces at t_in overlapping with p_out                    */
15     for Each p_in ∈ S[t_in].Pieces do
16         /* Get the corresponding piece from Tetris                   */
17         piece_in ← Tetris[p_in].Piece
18         /* Check if piece_out and piece_in overlaps                  */
19         if piece_out ∧ piece_in then
20             /* Add p_in to overlap_list                              */
21             overlap_list ← overlap_list ∪ pin
22         end
23     end
24     /* Switch the process between t_in and t_out                     */
25     check_list ← overlap_list
26     overlap_list ← φ
27     Swap(t_in, t_out)
28 end
29 return move_list_pieces, move_list_rows
```

Algorithm 9: The full cost function for DASA

Input : S, M, N, T, H, E, R, σ
Output: C, $M^{'}$, $assigned\_users$

$C \leftarrow 0$
/* Compute the sum of channel-gains for each user */
for $u \leftarrow 1$ **to** M **do**
  $h\_sum[u] \leftarrow Sum(H[u, 1..N])$
end
/* Initialize channel response and interference for all users */
$h_r[1..M, 1..T] \leftarrow 0$
$h_i[1..M, 1..T] \leftarrow h\_sum[1..M]$
/* Initialize the count of users assigned t time-slots 1 to T */
$assigned\_users[1..M] \leftarrow 0$
/* Compute channel response and interference for each user */
for $d \leftarrow 1$ **to** N **do**
  for $t \leftarrow 1$ **to** T **do**
    $u \leftarrow S[d, t]$
    $h_r[u, t] \leftarrow h_r[u, t] + H[u, d]$
    $h_i[u, t] \leftarrow h_i[u, t] - H[u, d]$
    $assigned\_users[u] \leftarrow assigned\_users[u] + 1$
  end
end
for $u \leftarrow 1$ **to** M **do**
  /* Compute the objective function */
  for $t \leftarrow 1$ **to** T **do**
    $\text{SINR} \leftarrow ER^2(h_r[u, t])^2/(ER^2(h_i[u, t])^2 + \sigma)$
    $C \leftarrow C + \log_2(1 + \text{SINR})$
  end
  /* Compute the number of users out of M assigned to time-slots 1 to T */
  if $assigned\_users[u] > 0$ **then**
    $M^{'} \leftarrow M^{'} + 1$
  end
end
return C, $M^{'}$, $assigned\_users$

Algorithm 10: The full cost function for TGSA

Input : S, $Tetris$, M, N, T, H, E, R, σ
Output: C, $M^{'}$, $assigned\_users$

```
1  C ← 0
2  /* Compute the sum of channel-gains for each user                */
3  for u ← 1 to M do
4  |  h_sum[u] ← Sum(H[u, 1..N])
5  end
6  /* Initialize channel response and interference for all users    */
7  h_r[1..M, 1..T] ← 0
8  h_i[1..M, 1..T] ← h_sum[1..M]
9  /* Initialize the count of users assigned t time-slots 1 to T   */
10 assigned_users[1..M] ← 0
11 /* Compute channel response and interference for each user      */
12 for t ← 1 to T do
13 |  for each p ∈ S[t].Pieces do
14 |  |  u ← Tetris[p].User
15 |  |  assigned_users[u] ← assigned_users[u] + 1
16 |  |  piece ← Tetris[p].Piece
17 |  |  for each d ∈ piece do
18 |  |  |  h_r[u, t] ← h_r[u, t] + H[u, d]
19 |  |  |  h_i[u, t] ← h_i[u, t] − H[u, d]
20 |  |  end
21 |  end
22 end
23 for u ← 1 to M do
24 |  /* Compute the objective function                             */
25 |  for t ← 1 to T do
26 |  |  SINR ← ER^2(h_r[u, t])^2/(ER^2(h_i[u, t])^2 + σ)
27 |  |  C ← C + log_2(1 + SINR)
28 |  end
29 |  /* Compute the number of users out of M assigned to time-slots
   |     1 to T                                                     */
30 |  if assigned_users[u] > 0 then
31 |  |  M' ← M' + 1
32 |  end
33 end
34 return C, M', assigned_users
```

Algorithm 11: The partial cost function for DASA

Input : S_{old}, S_{new}, C_{old}, $M^{'}_{old}$, $assigned\_users\_old$, N, T, t_x, t_y, u_x, u_y, H, E, R, σ

Output: C_{new}, $M^{'}_{new}$, $assigned\_users\_new$

$C_{new} \leftarrow C_{old}$; $M^{'}_{new} \leftarrow M^{'}_{old}$

$assigned\_users\_new \leftarrow assigned\_users\_old$

`/* If` u_x `and` u_y `are identical, then the cost will be the same */`

if $u_x = u_y$ **then return** C_{new}, $M^{'}_{new}$, $assigned\_users\_new$

`/* Initialization */`

InitPartialCost(N, T, t_x, t_y, u_x, u_y, H)

`/* Update users counts */`

if $t_x > T$ **then**

  $assigned\_users\_new[u_x] \leftarrow assigned\_users\_new[u_x] + 1$

  $assigned\_users\_new[u_y] \leftarrow assigned\_users\_new[u_y] - 1$

  if $assigned\_users\_new[u_x] = 1$ **then** $M^{'}_{new} \leftarrow M^{'}_{new} + 1$

  if $assigned\_users\_new[u_y] = 0$ **then** $M^{'}_{new} \leftarrow M^{'}_{new} - 1$

end

`/* Compute channel response and interference for each user */`

for $d \leftarrow 1$ **to** N **do**

  for $j \leftarrow 1$ **to** $|t_c|$ **do**

    $u_{old} \leftarrow S_{old}[d, t_c[j]]$; $u_{new} \leftarrow S_{new}[d, t_c[j]]$

    for $i \leftarrow 1$ **to** $|u_c|$ **do**

      if $u_{old} = u_c[i]$ **then**

        $h_r^{old}[i,j] \leftarrow h_r^{old}[i,j] + H[i,d]$; $h_i^{old}[i,j] \leftarrow h_i^{old}[i,j] - H[i,d]$

      end

      if $u_{new} = u_c[i]$ **then**

        $h_r^{new}[i,j] \leftarrow h_r^{new}[i,j] + H[i,d]$; $h_i^{new}[i,j] \leftarrow h_i^{new}[i,j] - H[i,d]$

      end

    end

  end

end

`/* Update the objective function */`

for $i \leftarrow 1$ **to** $|u_c|$ **do**

  for $j \leftarrow 1$ **to** $|t_c|$ **do**

    $\mathrm{SINR}_{old} \leftarrow ER^2(h_r^{old}[i,j])^2 / (ER^2(h_i^{old}[i,j])^2 + \sigma)$

    $\mathrm{SINR}_{new} \leftarrow ER^2(h_r^{new}[i,j])^2 / (ER^2(h_i^{new}[i,j])^2 + \sigma)$

    $C_{new} \leftarrow C_{new} + \log_2(1 + \mathrm{SINR}_{new}) - \log_2(1 + \mathrm{SINR}_{old})$

  end

end

return C_{new}, $M^{'}_{new}$, $assigned\_users\_new$

Algorithm 12: The procedure InitPartialCost for DASA

Input : N, T, t_x, t_y, u_x, u_y, H
Output: t_c, u_c, h_r^{old}, h_i^{old}, h_r^{new}, h_i^{new}

```
/* Initialize users lists                                          */
```
$u_c \leftarrow \{u_y, u_x\}$
```
/* Initialize time-slots lists                                     */
```
if $t_x \leq T$ **then**
```
    /* Both slots will affect the cost, but not the user's counts
       */
```
    $t_c \leftarrow \{t_y, t_x\}$
else
```
    /* Only t_y will affect the cost                               */
```
    $t_c \leftarrow \{t_y\}$
end
```
/* Compute the sum of channel-gains for the 2 user                 */
```
for $i \leftarrow 1$ **to** 2 **do**
    $h\_sum[i] \leftarrow Sum(H[u_c[i], 1..N])$
end
```
/* Initialize channel response and interference for the 2 users    */
```
$h_r^{old}[1..2, 1..2] \leftarrow 0$
$h_i^{old}[1..2, 1..2] \leftarrow h\_sum[1..2]$
$h_r^{new}[1..2, 1..2] \leftarrow 0$
$h_i^{new}[1..2, 1..2] \leftarrow h\_sum[1..2]$
return t_c, u_c, h_r^{old}, h_i^{old}, h_r^{new}, h_i^{new}

Algorithm 13: The partial cost function for TGSA

Input : S_{old}, S_{new}, C_{old}, $Tetris$, $assigned\_users\_old$, t_{out}, t_{in}, M, N, T, H, E, R, σ

Output: C_{new}, $M^{'}_{new}$, $assigned\_users\_new$

$C_{new} \leftarrow C_{old}$; $M^{'}_{new} \leftarrow M^{'}_{old}$
$assigned\_users\_new \leftarrow assigned\_users\_old$
InitPartialCost(M, N, T, t_{out}, t_{in}, H)
`/* Compute channel response and interference for each user */`
for $i \leftarrow 1$ **to** $|t_c|$ **do**
  if $t_c[i] \leq T$ **then**
    for each $p \in S_{old}[t_c[i]].Pieces$ **do**
      $u \leftarrow Tetris[p].User$
      $assigned\_users[u] \leftarrow assigned\_users[u] - 1$
      if $assigned\_users[u] = 0$ **then** $M^{'}_{new} \leftarrow M^{'}_{new} - 1$;
      $piece \leftarrow Tetris[p].Piece$
      for each $d \in piece$ **do**
        $h_r^{old}[u,i] \leftarrow h_r^{old}[u,i] + H[u,d]$; $h_i^{old}[u,i] \leftarrow h_i^{old}[u,i] - H[u,d]$
      end
    end
  end
  if $t_c[i] \leq T$ **then**
    for each $p \in S_{new}[t_c[i]].Pieces$ **do**
      $u \leftarrow Tetris[p].User$
      $assigned\_users[u] \leftarrow assigned\_users[u] + 1$
      if $assigned\_users[u] = 1$ **then** $M^{'}_{new} \leftarrow M^{'}_{new} + 1$;
      $piece \leftarrow Tetris[p].Piece$
      for each $d \in piece$ **do**
        $h_r^{new}[u,i] \leftarrow h_r^{new}[u,i] + H[u,d]$; $h_i^{new}[u,i] \leftarrow h_i^{new}[u,i] - H[u,d]$
      end
    end
  end
end
for $u \leftarrow 1$ **to** M **do**
  `/* Update the objective function */`
  if $t_c[i] \leq T$ **then**
    for $i \leftarrow 1$ **to** $|t_c|$ **do**
      $\text{SINR}_{old} \leftarrow ER^2(h_r^{old}[u,i])^2/(ER^2(h_i^{old}[u,i])^2 + \sigma)$
      $\text{SINR}_{new} \leftarrow ER^2(h_r^{new}[u,i])^2/(ER^2(h_i^{new}[u,i])^2 + \sigma)$
      $C_{new} \leftarrow C_{new} + \log_2(1 + \text{SINR}_{new}) - \log_2(1 + \text{SINR}_{old})$
    end
  end
end
return C_{new}, $M^{'}_{new}$, $assigned\_users\_new$

Algorithm 14: The procedure InitPartialCost for TGSA

Input : M, N, T, t_{out}, t_{in}, H
Output: t_c, h_r^{old}, h_i^{old}, h_r^{new}, h_i^{new}

```
/* Initialize time-slots lists                                          */
if t_out ≤ T then
    /* Both slots will affect the cost, but not the users counts        */
    t_c ← {t_in, t_out}
else
    /* Only t_in will affect the cost                                   */
    t_c ← {t_in}
end
/* Compute the sum of channel-gains for all users                       */
for u ← 1 to M do
    h_sum[u] ← Sum(H[u, 1..N])
end
/* Initialize channel response and interference for all users           */
h_r^old[1..M, 1..2] ← 0
h_i^old[1..M, 1..2] ← h_sum[1..M]
h_r^new[1..M, 1..2] ← 0
h_i^new[1..M, 1..2] ← h_sum[1..M]
return t_c, h_r^old, h_i^old, h_r^new, h_i^new
```

CHAPTER 5

NUMERICAL RESULTS AND DISCUSSION

As discussed in Chapter 4, we have developed two metaheuristics for the indoor resource allocation problem. The first algorithm is a direct application of the simulated annealing (DASA) and the second is a simulated annealing algorithm based on the Tetris-Game model (TGSA). In this chapter, we present the experimental results based on these algorithms.

5.1 Indoor RA Optimization

In our simulation of the proposed model for indoor environment, we have considered a room of size $12.0m \times 12.0m \times 3.0m$ with 16 LEDs on the ceiling spaced by $3.0m$ in both directions as illustrated in Fig. 5.1. The number of users is varied from 16 to 64 and are randomly distributed in the room. Table 5.1 shows the system and channel parameters used in our experiments [67]. The transmission power for each LED (P) is

35 dBm. The optical to electrical conversion coefficient (R) is 0.28 A/W. The system bandwidth (B) is 20 MHz. The noise density (N_0) is 10^{-22} W/Hz. The detector area of the receiver (A) and its FOV semiangle ($\Psi_{1/2}$) are $1cm^2$ and 60^o respectively. The transmitter semiangle ($\Phi_{1/2}$) is also 60^o.

We have implemented the proposed heuristic in MATLAB on 3.1 GHz Intel Core i7 processor with 16GB RAM. The parameters of the SA algorithm are as follows: $\alpha = 0.99$, $\beta = 1$, $T_0 = 1.0$, and $M_0 = 200$. Due to the stochastic nature of the SA algorithm, we did 100 trials on each experiment and average the result.

5.1.1 System Throughput Vs Number of Users

In this experiment, we have compared the system throughput with respect to the number of users for both the direct application of the SA algorithm (DASA) and the Tetris-Game model (TGSA). The results are presented in Fig. 5.2 which shows that TGSA outperforms DASA as the number of users increases. Note that we have fixed the number of time-slots per frame ($T = 12$), hence, the system throughput per user decreases as the number of users increases.

5.1.2 System Throughput Vs Number of Iterations

In this experiment, we have run the DASA and TGSA algorithms for a fixed number of iterations and compared their system throughput for 16 and 64 users. The best and current costs of the DASA and TGSA algorithms for 16 users are shown in Fig. 5.3 and 5.4 respectively. Both figures show the expected behavior of a standard simulated

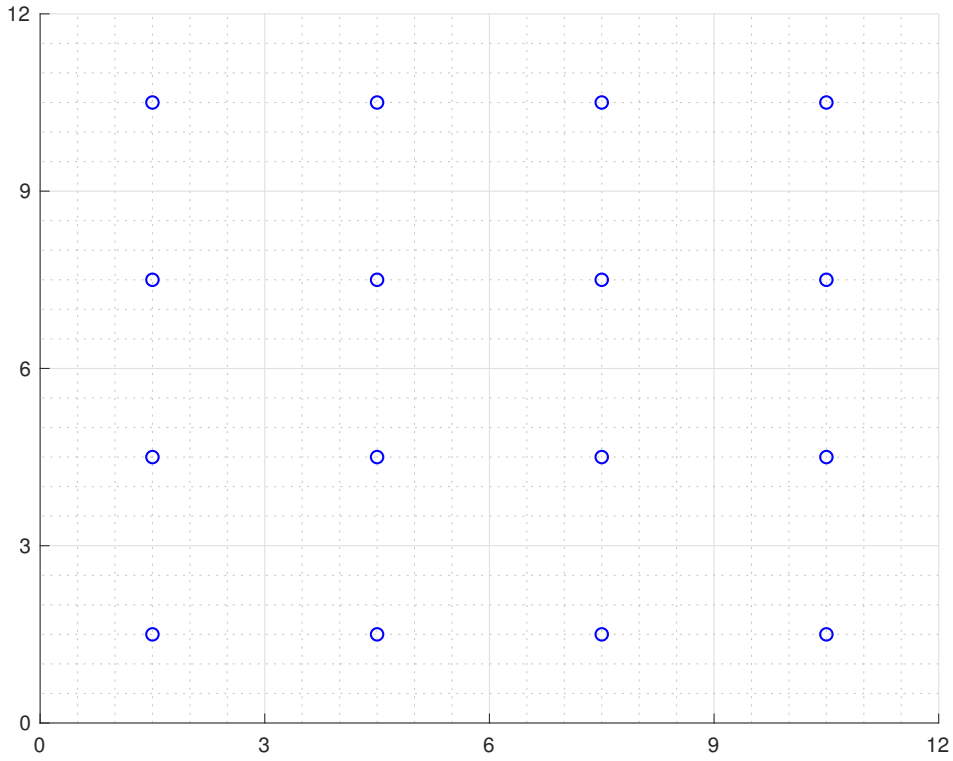

Figure 5.1: Distribution of LEDs in the room.

Table 5.1: System and channel parameters.

Parameter	**Symbol**	**Value**
Transmission Power for each LED	P	20 dBm
O/E Conversion Coefficient	R	0.28 A/W
System Bandwidth	B	5 MHz
Noise Density	N_0	10^{-22} W/Hz
Detector Area	(A)	$1cm^2$
Receiver FOV Semiangle	$\Psi_{1/2}$	60^o
Transmitter Semiangle	$\Phi_{1/2}$	60^o

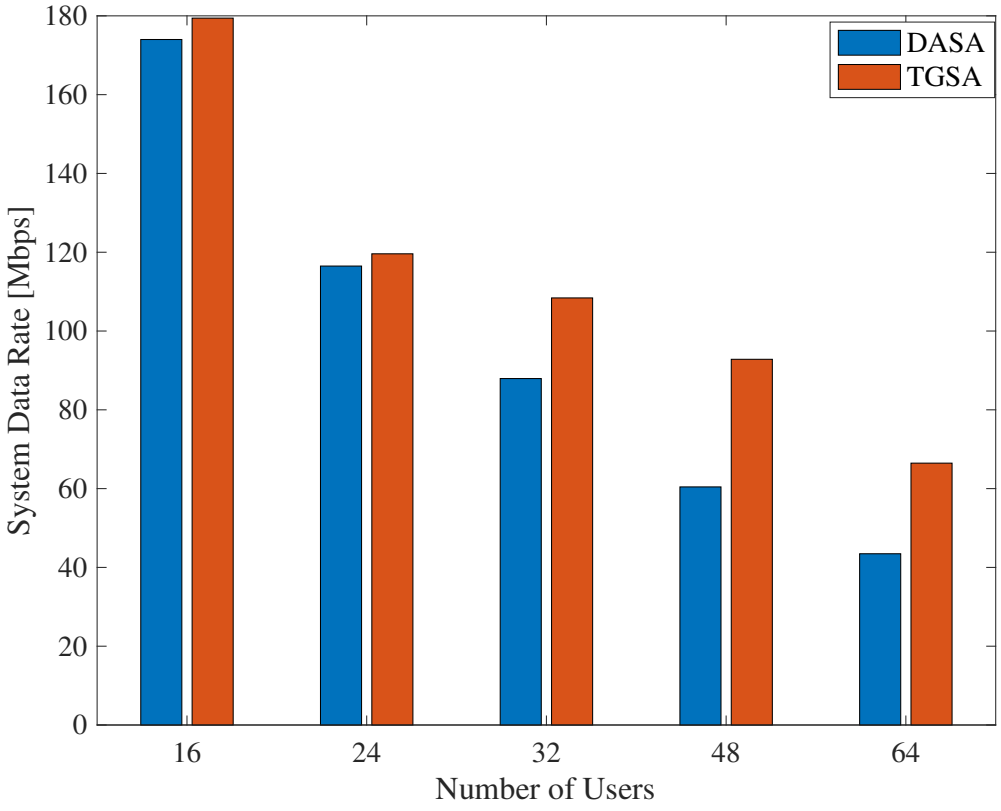

Figure 5.2: System throughput with respect to the number of users.

annealing algorithm with hill-climbing. Fig. 5.5 and 5.6 show comparisons between the best cost of the TGSA and that of DASA for 16 and 64 users respectively. These results show that the Tetris-Game model improves both the solution quality and the convergence rate.

5.2 Underwater RA Optimization

We consider the same setup as for the indoor environment except that users can move in the vertical direction. Assuming a pure water and LOS, the channel response can be calculated as given in equation (2.6). Fig. 5.7 shows a comparison between the best cost of TGSA and DASA for 16 users in underwater environment. The results show a similar behaviour of both algorithms where TGSA converges faster than DASA.

5.3 Results Discussion and Future Work

Our experiments using standard simulated annealing algorithm show that the final allocation of LEDs takes a form similar to the arrangement of the Tetris pieces. This is illustrated in in Table 5.2 for an indoor environment with nine LEDs ($N = 9$), eight users ($M = 8$) and eight time-slots per frame ($T = 8$). The channel-gains of the users are shown in Table 5.3. This observation is consistent with the results obtained in Section 5.1.2 which show clearly that TGSA converges faster than DASA as it operates on groups of assignments rather than individual ones.

In future work, we are planning to investigate more meta-heuristics and optimize the proposed algorithms to satisfy the the channel coherence time. In addition to that,

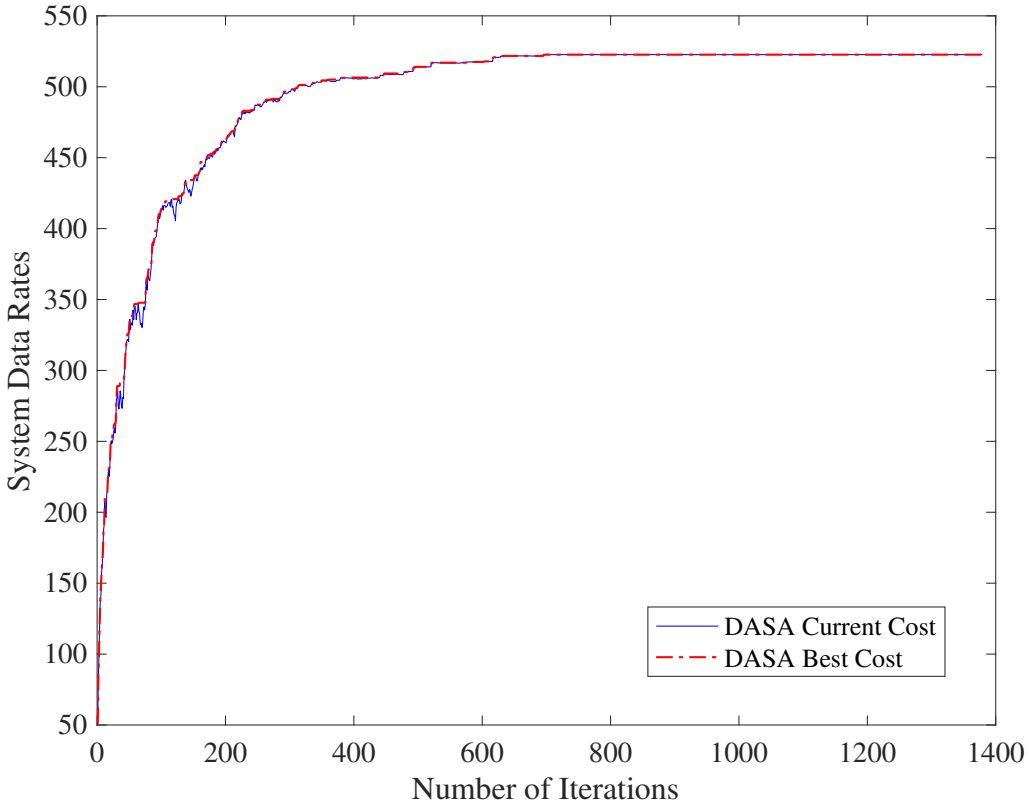

Figure 5.3: Best and current costs of the DASA for 16 users.

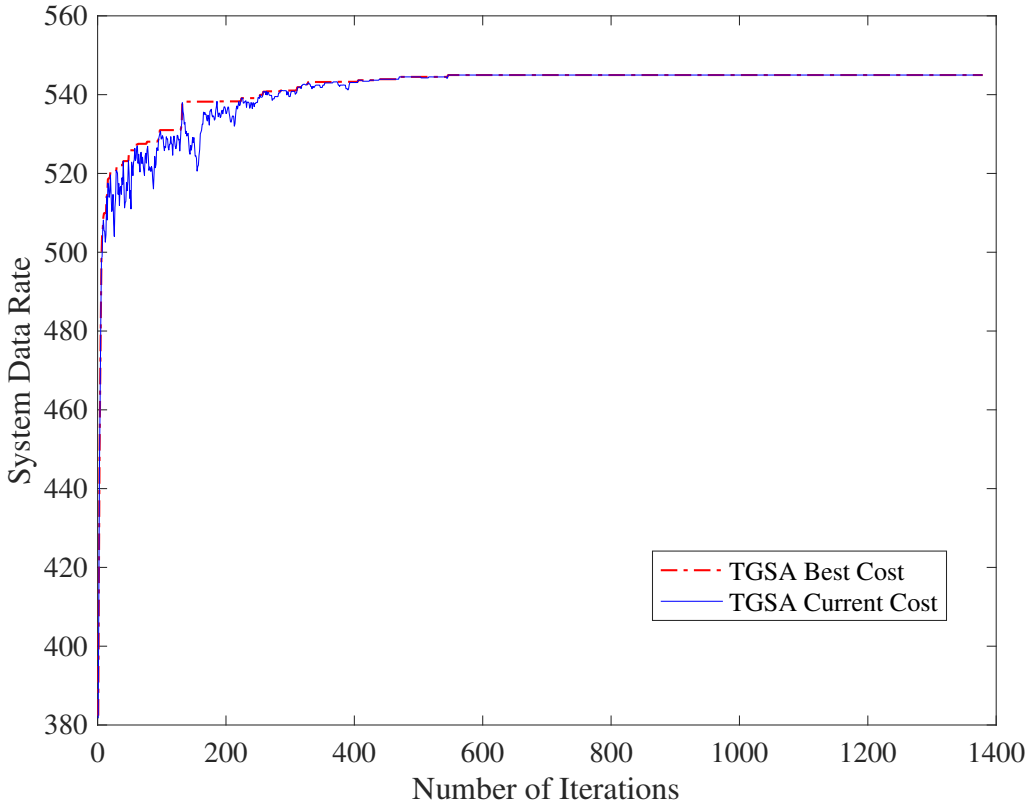

Figure 5.4: Best and current costs of the TGSA for 16 users.

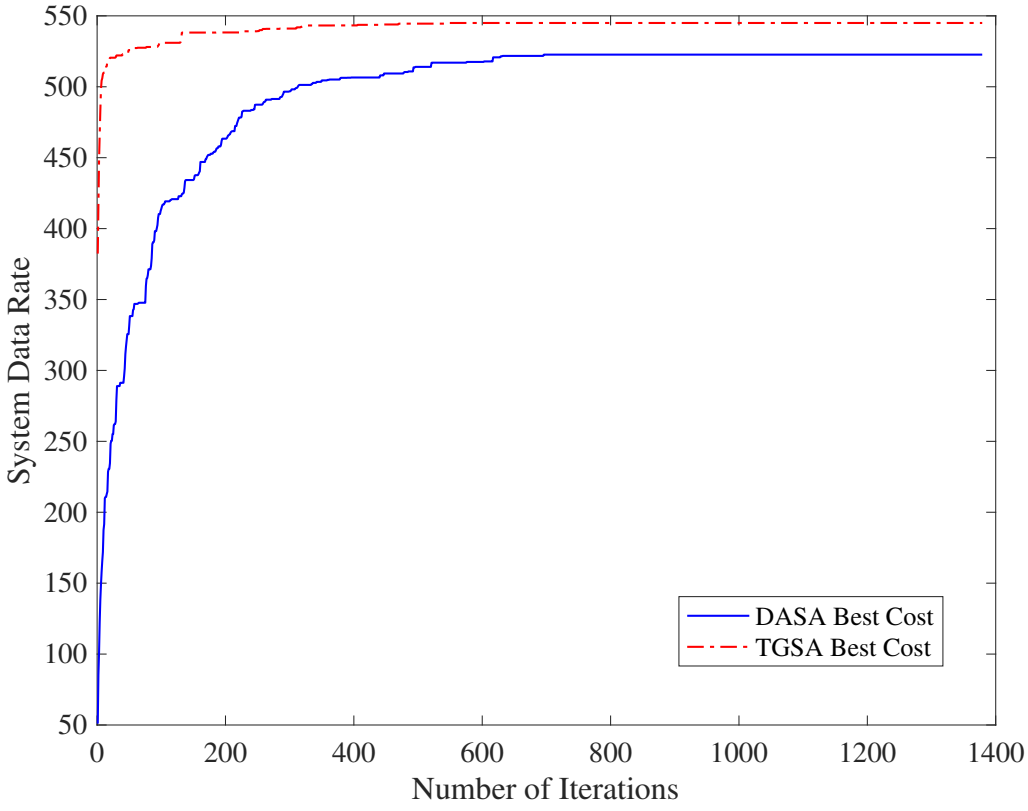

Figure 5.5: Comparison of the best costs of TGSA and DASA for 16 users.

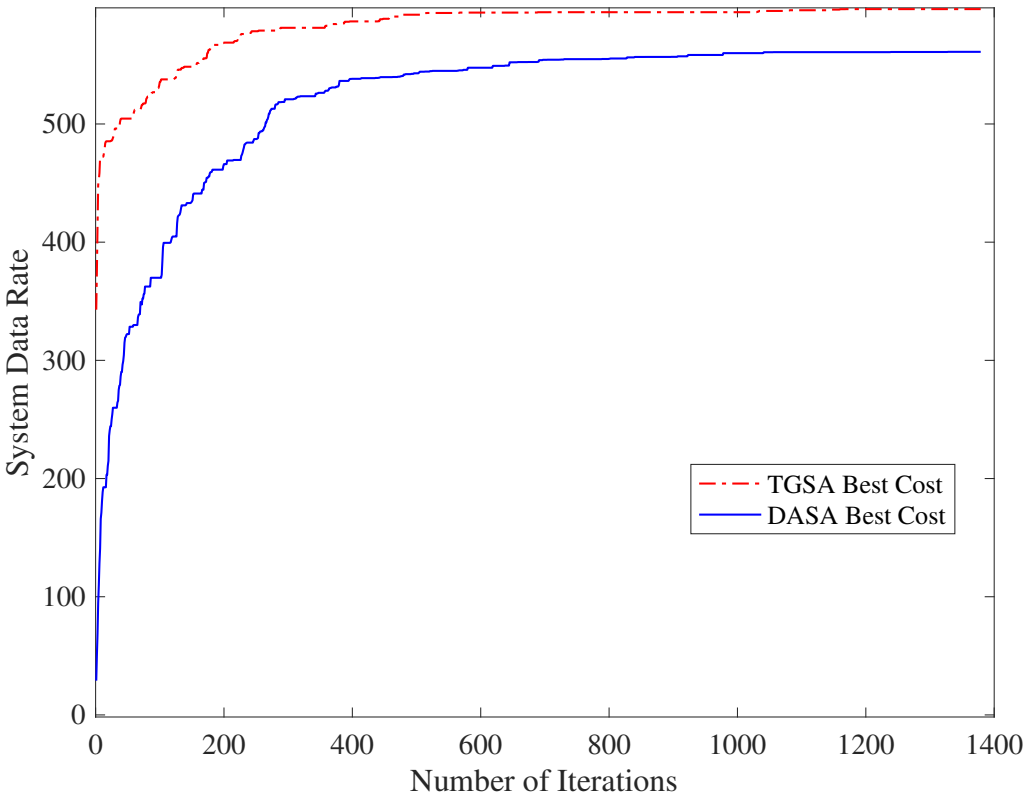

Figure 5.6: Comparison of the best costs of TGSA and DASA for 64 users.

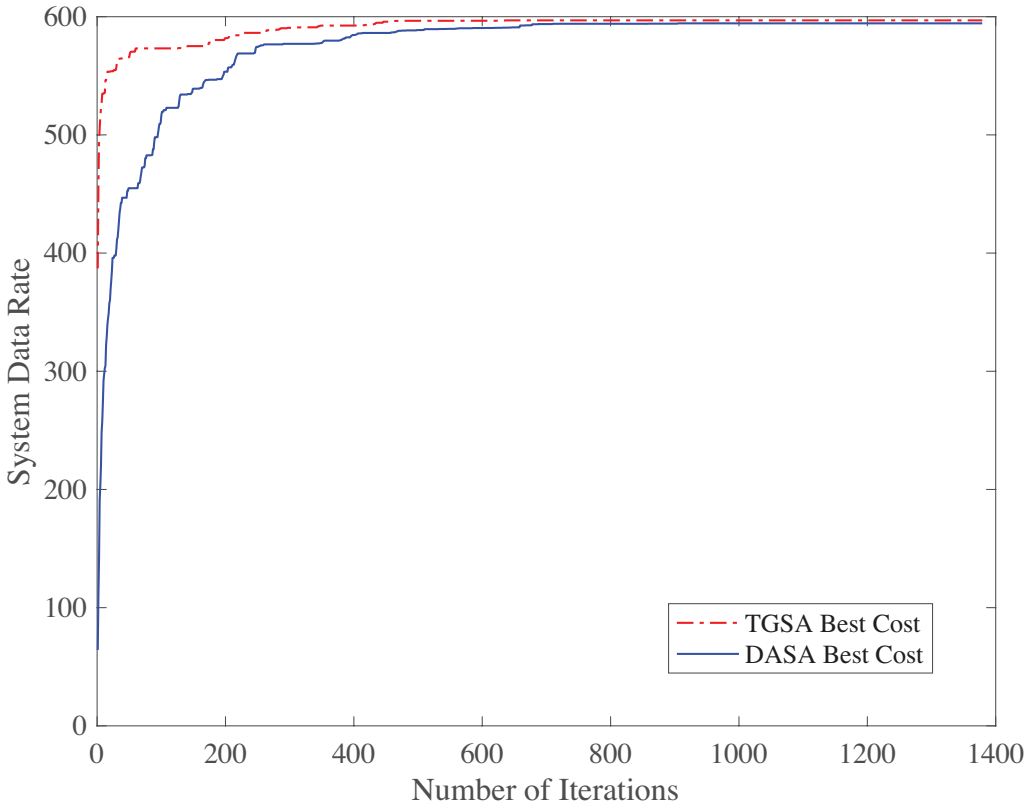

Figure 5.7: Comparison of the best costs of TGSA and DASA for 16 users in underwater environment.

we will develop optimization heuristics for different types of water for the underwater VLC problem (In our experiments, we considered only pure water types).

Table 5.2: Allocation result from a standard simulated annealing implementation for 8 users, 9 LEDs and 8 time-slots per frame ($M = 8, N = 9, T = 8$).

	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
D1	6	6	6	6	6	6	6	6
D2	6	6	6	6	6	6	6	6
D3	4	4	4	4	4	4	4	4
D4	1	5	1	1	1	1	1	3
D5	1	8	1	1	1	1	1	8
D6	4	4	4	4	4	4	4	4
D7	1	8	1	1	1	1	1	8
D8	1	8	1	1	1	1	1	8
D9	7	8	7	7	7	2	7	8

Table 5.3: Channel-gains of the example shown in Table 5.2.

	D1	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
U1	0	0	0	0.1012	0.1218	0	0.2165	0.2856	0
U2	0	0	0	0	0.0861	0.2135	0	0.1113	0.3247
U3	0.0517	0.5205	0.1312	0	0.1004	0.0487	0	0	0
U4	0	0	0.1268	0	0	0.2453	0	0	0
U5	0	0	0	0	0.18	0.0791	0.0426	0.3649	0.1227
U6	0.2732	0.214	0	0	0	0	0	0	0
U7	0	0	0	0	0.0414	0.05	0	0.2226	0.3539
U8	0	0	0	0	0.064	0.0392	0.0447	0.4836	0.1557

APPENDIX A

SUPPLEMENTARY MATERIAL

A.1 MATLAB Code: Direct Application Simulated Annealing

```
%% Parameters
global L W H LED PD M N T S A h h_sum E R sigma BW X;

% TDMA parameters
M = 8;                      % number of users
N = 9;                      % number of LEDs
T = 8;                      % number of time-slots
S = N * T;                  % total number of slots
A = floor(S/M);             % number of slots per user
X = 16;
L = 9;                      % room length
W = 9;                      % room width
H = 3;                      % room height

E_dBm = 20;                 % symbol energy in dBm
E = db2pow(E_dBm-30);       % symbol energy in Watt 10log_10(y)
R = 0.28;                   % optical to electrical conversion
                            % coefficient
BW = 5e+6;                  % system bandwidth in Hz (per LED)
sigma = (1e-22) * BW;       % noise power in Watt
```

```
squares = sqrt(N);          % number of squares in each
                            % row/column in the room
step = L/squares;           % separation distance between
                            % two LEDs
start = step/2;             % starting location of the 1st LED
x = start:step:L;           % x coordinates of the LEDs
y = x;                      % y coordinates of the LEDs

% compute the x-y-z coorinates of the LEDs
LED = zeros(N,3);
LED(:,3) = H;
LED(:,1:2) = combvec(x,y)';

% generate x-y-z coordinates of the users (photo-diodes)
PD = [  round(rand(M,1)*L,2) , ...
        round(rand(M,1)*W,2) , ...
        round(0.7*ones(M,1),2) ];

% get channel responses
h = compute_channel_responses();

load('vlc8.mat', 'PD', 'h');

h_sum = sum(h,2);

%% Simulated Annealing Function
solution = generate_random_solution();
% SA Parameters
S0 = solution;                          % initial solution
T0 = 1;                                 % initial temperature
Tf = 1.0e-3;                            % final temperature
alpha = 0.995;                          % cooling rate
beta = 1;                               % constant
Tp = 200;                               % time until the next
                                        % parameter
                                        % update
Tc = T0;                                % current temperature
CurS = S0;                              % current solution
BestS = S0;                             % best solution seen
CurC = CurS.cost * CurS.count / M;      % current cost
BestC = CurC;                           % best cost seen
MaxIter = count_iterations(T0,Tf,alpha);
DASA_CC = zeros(1,MaxIter);
DASA_BB = zeros(1,MaxIter);
Iter = 1;
```

```
while Tc > Tf
    count = 0;
    while count < Tp
        NewS = neighbor(CurS);
        NewC = NewS.cost * NewS.count / M;
        DeltaC = NewC - CurC;
        if DeltaC > 0
            CurS = NewS;
            CurC = NewC;
            if NewC > BestC
                BestS = NewS;
                BestC = NewC;
            end
        elseif rand <= exp(DeltaC/Tc)
            CurS = NewS;
            CurC = NewC;
        end
        count = count + 1;
    end
    DASA_CC_16(Iter) = CurC;
    DASA_BB_16(Iter) = BestS.cost;
    Iter = Iter + 1;


Tc = alpha * Tc;
    Tp = beta * Tp;
end
figure(1);
plot_solution(BestS, 0.0, BestC, 0.0);
figure(2);
plot(DASA_CC);
figure(3);
plot(DASA_BB);

%% Count Number of Iterations
function n = count_iterations(T0,Tf,alpha)
n = 0;
Tc = T0;
while Tc > Tf
    Tc = Tc * alpha;
    n = n + 1;
end
end

%% Compute Cost (Fitness)
```

```
function f = cost_partial(s_old, s_new, tx, ty, ux, uy)
% computes the cost (fitness) of a given solution
global N T h h_sum E R sigma;

% if the user is the same in
% both slots, then no change in the cost
if ux == uy
    f = 0;
    return;
end

% initialize the channel responses/
% interferences to be removed from the
% original solution
RHSux = zeros(2,1);
RHIux = [h_sum(ux); h_sum(ux)];
RHSuy = zeros(2,1);
RHIuy = [h_sum(uy); h_sum(uy)];

% initialize the channel responses
% /interferences to be added to the
% modified solution
AHSux = zeros(2,1);
AHIux = [h_sum(ux); h_sum(ux)];
AHSuy = zeros(2,1);
AHIuy = [h_sum(uy); h_sum(uy)];

% initialize SINR for channel responses to
% be removed from the original
% solution
RSINRux = zeros(2,1);
RSINRuy = zeros(2,1);

% initialize SINR for channel responses
% to be added to the modified
% solution
ASINRux = zeros(2,1);
ASINRuy = zeros(2,1);

for d = 1:N
    % get users assignments in
    % tx and ty for the d^th led
    u_old = s_old(d,:);
    u_new = s_new(d,:);
```

```
        % recalculation for time slot tx will
        % be done if and only if tx <= T and tx ~= ty
        if tx <= T && tx ~= ty
            if u_old(1) == ux
                RHSux(1) = RHSux(1) + h(ux,d);
                RHIux(1) = RHIux(1) - h(ux,d);
            elseif u_old(1) == uy
                RHSuy(1) = RHSuy(1) + h(uy,d);
                RHIuy(1) = RHIuy(1) - h(uy,d);
            end

            if u_new(1) == ux
                AHSux(1) = AHSux(1) + h(ux,d);
                AHIux(1) = AHIux(1) - h(ux,d);
            elseif u_new(1) == uy
                AHSuy(1) = AHSuy(1) + h(uy,d);
                AHIuy(1) = AHIuy(1) - h(uy,d);
            end
        end

        % recalculation for time slot
        % ty will be done in all cases
        if u_old(2) == ux
            RHSux(2) = RHSux(2) + h(ux,d);
            RHIux(2) = RHIux(2) - h(ux,d);
        elseif u_old(2) == uy
            RHSuy(2) = RHSuy(2) + h(uy,d);
            RHIuy(2) = RHIuy(2) - h(uy,d);
        end

        if u_new(2) == ux
            AHSux(2) = AHSux(2) + h(ux,d);
            AHIux(2) = AHIux(2) - h(ux,d);
        elseif u_new(2) == uy
           AHSuy(2) = AHSuy(2) + h(uy,d);
           AHIuy(2) = AHIuy(2) - h(uy,d);
        end

    end

    % compute SINR for
    % removed channel responses
    RSINRux(1) = E * R^2 * RHSux(1)^2 /...
    (E * R^2 * RHIux(1)^2 + sigma);
    RSINRux(2) = E * R^2 * RHSux(2)^2 /...
```

```
(E * R^2 * RHIux(2)^2 + sigma);
RSINRuy(1) = E * R^2 * RHSuy(1)^2 /...
(E * R^2 * RHIuy(1)^2 + sigma);
RSINRuy(2) = E * R^2 * RHSuy(2)^2 /...
(E * R^2 * RHIuy(2)^2 + sigma);

% compute SINR for added channel responses
ASINRux(1) = E * R^2 * AHSux(1)^2 /...
(E * R^2 * AHIux(1)^2 + sigma);
ASINRux(2) = E * R^2 * AHSux(2)^2 /...
(E * R^2 * AHIux(2)^2 + sigma);
ASINRuy(1) = E * R^2 * AHSuy(1)^2 /...
(E * R^2 * AHIuy(1)^2 + sigma);
ASINRuy(2) = E * R^2 * AHSuy(2)^2 /...
(E * R^2 * AHIuy(2)^2 + sigma);

% compute cost incremental value
f = log2(1 + ASINRux(1)) + ...
    log2(1 + ASINRux(2)) + ...
    log2(1 + ASINRuy(1)) + ...
    log2(1 + ASINRuy(2)) - ...
    log2(1 + RSINRux(1)) - ...
    log2(1 + RSINRux(2)) - ...
    log2(1 + RSINRuy(1)) - ...
    log2(1 + RSINRuy(2));

end

%% Compute Cost (Fitness)
function f = cost_full(solution)
% computes the cost (fitness) of a given solution
global M N T h h_sum E R sigma;

% initialize SINR
SINR = zeros(M,T);

overall_assigned_users = zeros(M,1);
for t = 1:T
    % initialize channel responses and
    % interference in the t^th time-slot
    hs = zeros(M,1);
    hi = h_sum;

    % initialize set of assigned users
    current_assigned_users = false(M,1);
```

```
    for d = 1:N
        % get user assignment
        u = solution.assignment(d,t);

        % skip unassigned slots
        if u == 0
            hi(:) = hi(:) - h(:,d);
            continue;
        end

        % compute channel responses and
        % interference for the u^th user
        hs(u) = hs(u) + h(u,d);
        hi(u) = hi(u) - h(u,d);

        % add user to the assigned users set
        current_assigned_users(u) = true;
        overall_assigned_users(u) = 1;
    end
    % compute SINR for users assigned
    % in the t^th time slots
    for u = 1:M
        if current_assigned_users(u)
            SINR(u,t) = E * R^2 * hs(u)^2 / ...
            (E * R^2 * hi(u)^2 + sigma);
        end
    end
end
% compute data rate per user
DR = sum(log2(1+SINR),2);
% compute the total data rate
f = sum(DR, 'all');
end


%% Create Neighbor
function s = neighbor(solution)
% creates a new solution by exchanging two assignments selected random
global S X;
s = solution;

% select two different slots randomly
si = 0;
sj = 0;
```

```
while si == sj
    si = randi(X*S);
    sj = randi(S);
end

ss = size(s.assignment);
[di, ti] = ind2sub(ss, si);
[dj, tj] = ind2sub(ss, sj);

ui = s.assignment(si);
uj = s.assignment(sj);

% get assignments in ti and tj before exchange
s_old = [s.assignment(:,ti) s.assignment(:,tj)];

% if si is not in the first T time slots,
% then update assigned users count
if si > S
    % if ui is not assigned any slot
    % in the first T time slots, then
    % increment users count
    if s.users(ui) == 0
        s.count = s.count + 1;
    end

    % update users assignments count
    s.users(ui) = s.users(ui) + 1;
    s.users(uj) = s.users(uj) - 1;

    % if uj is not assigned to any
    % slot in the first T time slots, then
    % decrement users count
    if s.users(uj) == 0
        s.count = s.count - 1;
    end

end

% interchange these two slots
temp = s.assignment(si);
s.assignment(si) = s.assignment(sj);
s.assignment(sj) = temp;

s_new = [s.assignment(:,ti) s.assignment(:,tj)];
s.cost = s.cost + cost_partial(s_old, s_new, ...
```

```
ti, tj, ui, uj);
end

%% Generate Random Solution
% generates a random solution with equal time slots for each user
function s = generate_random_solution()
global M N T A S X;

% intialize solution structure
% assignment: matrix of int
%   user assignment matrix (LEDs x Time Slots)
% cost: float
%   cost of the solution
% users: vector of int
%   number of assignments to each user in the first T slots
% count: int
%   number of users out of M assigned to the first T slots
s = struct('assignment', zeros(N,X*T), ...
    'cost', 0.0, ...
    'users', zeros(M,1), ...
    'count', 0);

% generate A random samples of users
users_samples = zeros(X*A,M);
for i = 1:X*A
    users_samples(i,:) = randsample(M,M)';
end

% assign these random samples to the NxT slots
for i = 1:X*S

    % get a random user
    user = users_samples(i);

    % increment assigned users count in all slots <= T
    if i <= S
        if s.users(user) == 0
            s.count = s.count + 1;
        end
        s.users(user) = s.users(user) + 1;
    end

    % assign user
    s.assignment(i) = user;
```

```
end
s.cost = cost_full(s);
end

%% Plot Room
% plot the room with LEDs and PDs
function f = plot_room()
global M PD LED L W;
shg
f = 0;
% plot LEDs' locations
scatter(LED(:,1), LED(:,2), 'b');

% plot PDs' locations
hold on;
scatter(PD(:,1), PD(:,2), 'r');
grid minor;
axis([0 L 0 W])

% disply users' labels
users = (1:M)';
labels = num2str(users,'%d');
text(PD(:,1), PD(:,2), labels, ...
'horizontal','left', 'vertical','bottom');

drawnow;
end

%% Plot Solution
% plot solution as a heat-map
function f = plot_solution(solution,CurC, BestC, t)
global T;
shg
f = 0;
heatmap(solution.assignment(:,1:T));
str = sprintf('Current: %f, Best: %f, ...
Temp: %f',CurC, BestC, t);
%title(str);
drawnow;
end
```

A.2 MATLAB Code: Tetris Game Model Simulated Annealing

```
%% Parameters
global L W H LED PD M N T S A h E R sigma tetris BW h_sum;

% TDMA parameters
M = 16;                     % number of users
N = 16;                     % number of LEDs
T = 12;                     % number of time-slots
S = N * T;                  % total number of slots
%A = floor(S/M);            % number of slots per user
A = 18;                     % number of pieces of each tetris
                            % combination
L = 12;                     % room length
W = 12;                     % room width
H = 3;                      % room height
E_dBm = 20;                 % symbol energy in dBm
E = db2pow(E_dBm-30);       % symbol energy in Watt10log_10(y)
R = 0.28;                   % optical to electrical
                            % conversion coefficient
BW = 5e+6;                  % system bandwidth in Hz (per LED)
sigma = (1e-22) * BW;       % noise power in Watt
squares = sqrt(N);          % number of squares in each
                            % row/column in the room
step = L/squares;           % separation distance
                            % between two LEDs
start = step/2;             % starting location of the 1st LED
x = start:step:L;           % x coordinates of the LEDs
y = x;                      % y coordinates of the LEDs

% compute the x-y-z coorinates of the LEDs
LED = zeros(N,3);
LED(:,3) = H;
LED(:,1:2) = combvec(x,y)';

% generate x-y-z coordinates of the users (photo-diodes)
PD = [  round(rand(M,1)*L,2) , ...
        round(rand(M,1)*W,2) , ...
        round(0.7*ones(M,1),2) ];

% get channel responses
h = compute_channel_responses();

load('vlc16.mat', 'PD', 'h');

h_sum = sum(h,2);
```

```
% compute tetris pieces
tetris = compute_tetris_pieces();

%% Simulated Annealing
% generate an inital random solution
solution = generate_random_solution();

% SA Parameters
S0 = solution;      % initial solution
T0 = 1;             % initial temperature
Tf = 1.0e-3;        % final temperature
alpha = 0.995;      % cooling rate
beta = 1;           % constant
Tp = 200;           % time until the next parameter update

Tc = T0;             % current temperature
CurS = S0;           % current solution
BestS = S0;          % best solution seen so far
CurC = CurS.cost * CurS.users_count / M;    % current cost
BestC = CurC;        % best cost seen so far

MaxIter = count_iterations(T0,Tf,alpha);
TGSA_CC = zeros(1,MaxIter);
TGSA_BB = zeros(1,MaxIter);
Iter = 1;

while Tc > Tf
    count = 0;
    while count < Tp
        NewS = neighbor(CurS);
        NewC = NewS.cost * NewS.users_count / M;
        DeltaC = (NewC - CurC);
        if DeltaC > 0
            CurS = NewS;
            CurC = NewC;
            if NewC > BestC
                BestS = NewS;
                BestC = NewC;
            end
        elseif rand <= exp(DeltaC/Tc)
            CurS = NewS;
            CurC = NewC;
        end
        count = count + 1;
    end
```

```
    %plot_solution(CurS,CurS.cost, BestS.cost, Tc);

    TGSA_CC(Iter) = CurC;
    TGSA_BB(Iter) = BestC;
    Iter = Iter + 1;
    Tc = alpha * Tc;
    Tp = beta * Tp;
end
figure(1);
plot_solution(BestS, 0.0, BestC, 0.0);
figure(2);
plot(TGSA_CC);
figure(3);
plot(TGSA_BB);

%% Count Number of Iterations
function n = count_iterations(T0,Tf,alpha)
n = 0;
Tc = T0;
while Tc > Tf
    Tc = Tc * alpha;
    n = n + 1;
end
end
%% Neighbor
function s = neighbor(solution)
% creates a new solution by removing
% one piece randomly from the given
% solution and assigning it again in a random order
global tetris N T;
s = solution;
n = length(s.slots);
%
% step(1): select two random time
% slots t_out and t_in and random piece
%
% select two random time slots
t_out = 0;
t_in = 0;
while t_out == t_in
    t_out = randi(n);
    t_in = randi(T);
end
% select random piece
index = randi(s.slots(t_out).pieces_count);
```

```
% store old slots
s_old = [s.slots(t_out) s.slots(t_in)];

% store initial t_out and t_in time-slot
t_out_init = t_out;
t_in_init = t_in;

% initialize overlap list
overlap_list = zeros(1,N);

% initialize check list
check_list = zeros(1,N);
check_cnt = 1;
check_list(check_cnt) = index;

% initialize move list
move_list_pieces = zeros(1,N);
move_list_slots = zeros(1,N);
move_cnt = 0;

%
% step(2): move p_out to t_in and move
% overlapping pieces between the slots
%
while(check_cnt > 0)
    overlap_cnt = 0;
    inserted_overlaps = false(1,N);
    % for each piece in the check list
    % find overlaping pieces
    for i = 1:check_cnt
        index_out = check_list(i);
        % get piece assignment value
        p_out = s.slots(t_out).pieces(index_out);
        piece_out = tetris(p_out).piece;

        % remove the piece from t_out slot
        s.slots(t_out).assignment = ...
        and(s.slots(t_out).assignment, ~piece_out);
        s.slots(t_out).pieces(index_out) = 0;
        s.slots(t_out).pieces_count = ...
        s.slots(t_out).pieces_count - 1;
        % add the piece to the move list,
        % so it will be placed at t_in
        move_cnt = move_cnt + 1;
```

```
            move_list_pieces(move_cnt) = p_out;
            move_list_slots(move_cnt) = t_in;

            % check for overlap between pieces
            % in the check list and those in
            % t_in
            for index_in = 1:s.slots(t_in).pieces_count
                % get piece detalis

                p_in = s.slots(t_in).pieces(index_in);
                piece_in = tetris(p_in).piece;

                % evaluate overlapping between the two pieces
                overlap = any(and(piece_in, piece_out));

                % check if the two pieces overlap
                if (overlap)
                    % add the index of overlapping
                    % piece to overlap list
                    if ~inserted_overlaps(index_in)
                        inserted_overlaps(index_in) = true;
                        overlap_cnt = overlap_cnt + 1;
                        overlap_list(overlap_cnt) = index_in;
                    end
                end
            end
        end
        % remove pices from t_out
        remaining_pieces = ...
        find(s.slots(t_out).pieces > 0);
        n = s.slots(t_out).pieces_count;
        s.slots(t_out).pieces(1:n) = ...
        s.slots(t_out).pieces(remaining_pieces);
        s.slots(t_out).pieces(n+1:end) = 0;
        % update check list
        check_list = overlap_list;
        check_cnt = overlap_cnt;

        % swap the two time slots
        t = t_in;
        t_in = t_out;
        t_out = t;
    end

    t_out = t_out_init;
```

```
t_in = t_in_init;

%
% step(3): place pieces in move list
% in their proper time slots
%
% for each element in the move list
for i = 1:move_cnt
     % get element detalis
    p = move_list_pieces(i);
    t = move_list_slots(i);

    % get the piece details from the tetris list
    piece = tetris(p).piece;
    u = tetris(p).user;

    % update users count if t_out > T
    if t_out > T
        if t <= T
            if s.users(u) == 0
                s.users_count = s.users_count + 1;
            end
            s.users(u) = s.users(u) + 1;
        else
            s.users(u) = s.users(u) - 1;
            if s.users(u) == 0
                s.users_count = s.users_count - 1;
            end
        end
    end

    % place the piece
    s.slots(t).assignment = or(piece,s.slots(t).assignment);
    n = s.slots(t).pieces_count;
    s.slots(t).pieces(n+1) = p;
    s.slots(t).pieces_count = n + 1;
end

% store new slots
s_new = [s.slots(t_out) s.slots(t_in)];

%
% step(4): remove initial t_out time-slot if it is empty
%
if s.slots(t_out).pieces_count == 0
```

```
    % remove empty time slots
    s.slots(t_out) = [];
end

f = cost_partial(s_old, s_new, [t_out t_in]);
%disp(f);
%disp(s.cost);
s.cost = s.cost + f;
%disp(s.cost);
end

%% Generate Random Solution
function solution = generate_random_solution()
% generate a random solution
global N M T tetris A;

% get the total number of pieces in the tetris
n = length(tetris);

% solution (assignment) structure
% assignment: logical vector size N
%   merges all pieces into to a time-slot
% pieces: int vector of size N
%   contains the indeces of assignment pieces
% pieces_count: int
%   counts number of used pieces
% users: int vector of size M
%   number of assignments to each user in the first T slots
% users_count: int
%   number of users out of M assigned to the first T slots
% cost: float
%   cost (fitness) of the solution

solution = struct('slots', repmat(struct(...
    'assignment', false(1,N), ...
    'pieces', zeros(1,N), ...
    'pieces_count', 0), n*A, 1), ...
    'users', zeros(1,M), ...
    'users_count', 0, ...
    'cost', 0.0);

% generate a random sample from the tetris
tetris_sample = randsample(n*A,n*A);
tetris_sample = mod(tetris_sample,n) + 1;
n = n * A;
```

```
for i = 1:length(tetris_sample)
    % get the index of the i^th sample (piece)
    index = tetris_sample(i);

    % get the piece from the tetris
    piece = tetris(index).piece;

    for t = 1:n
        % get the assignment in the t^th
        % solution (t^th time-slot)
        assignment = solution.slots(t).assignment;

        % check overlapping between the i^th
        % piece and t^th time-slot
        overlap = and(piece,assignment);

        % if no overlap, then do the following
        if ~overlap
            % merge the piece with the
            % current assignment
            solution.slots(t).assignment =...
            or(piece,assignment);

            % add the index of the assigned piece
            m = solution.slots(t).pieces_count;
            solution.slots(t).pieces(m+1) = index;
            solution.slots(t).pieces_count = m + 1;
            break;
        end
    end
end

empty_slots = zeros(n,1);
k = 0;

for t = 1:n
    if solution.slots(t).pieces_count == 0
        % if time-slot is empty, then add it to the empty-slots list
        k = k + 1;
        empty_slots(k) = t;
    end
end

% delete empty time-slots from the solution
```

```
solution.slots(empty_slots(1:k)) = [];

for t = 1:T
    % update users count for current time-slot
    for i = 1:solution.slots(t).pieces_count
        p = solution.slots(t).pieces(i);
        u = tetris(p).user;
        if solution.users(u) == 0
            solution.users_count = solution.users_count + 1;
        end
        solution.users(u) = solution.users(u) + 1;
    end
end

solution.cost = cost(solution);
end

%% Compute Solution Cost (Fitness)
function f = cost_partial(s_old, s_new, t)
% computes the cost (fitness) of a given solution
global tetris M T h E R sigma h_sum;

% initialize channel responses to be removed from t_out and t_in
RHs = zeros(M,2);
RHi = ones(M,2) .* h_sum;

% initialize channel responses to be added to t_out and t_in
AHs = zeros(M,2);
AHi = ones(M,2) .* h_sum;

% initialize SINR to be removed from t_out and t_in
RSINR = zeros(M,2);

% initialize SINR to be added to t_out and t_in
ASINR = zeros(M,2);

% initialize added/removed users
users_removed = false(M,2);
users_added = false(M,2);

% compute channel responses of to
% be removed and added
for i = 1:2
    % compute channel responses to be
    % removed from t_out and t_in
```

```
for j = 1:s_old(i).pieces_count
    % get piece details
    p = s_old(i).pieces(j);
    piece = tetris(p).piece;
    u = tetris(p).user;

    % update removed users
    users_removed(u,i) = true;

    % determine active-leds in the
    % piece assignment
    active_leds = find(piece > 0);

    % determine the sum of channel
    % responses of active_leds
    hh = sum(h(u,active_leds));

    % remove channel responses if and only if t <= T
    if t(i) <= T
        RHs(u,i) = RHs(u,i) + hh;
        RHi(u,i) = RHi(u,i) - hh;
    end
end

% compute channel responses to
% be added to t_out and t_in
for j = 1:s_new(i).pieces_count
    % get piece details
    p = s_new(i).pieces(j);
    piece = tetris(p).piece;
    u = tetris(p).user;

    % update added users
    users_added(u,i) = true;

    % determine active-leds in the piece assignment
    active_leds = find(piece > 0);

    % determine the sum of channel
    % responses of active_leds
    hh = sum(h(u,active_leds));

    % add channel responses if and only if t <= T
    if t(i) <= T
        AHs(u,i) = AHs(u,i) + hh;
```

```
            AHi(u,i) = AHi(u,i) - hh;
        end
    end
end


% compute incremental cost
f = 0;
for u = 1:M
    for i = 1:2
        % compute SINR to be removed
        if users_removed(u,i)
            RSINR(u,i) = E * R^2 * RHs(u,i)^2 / ...
            (E * R^2 * RHi(u,i)^2 + sigma);
        end
        % compute SINR to be added
        if users_added(u,i)
            ASINR(u,i) = E * R^2 * AHs(u,i)^2 /...
            (E * R^2 * AHi(u,i)^2 + sigma);
        end
        f = f + log2(1 + ASINR(u,i)) - log2(1 + RSINR(u,i));
    end
end

%unassigned_users = M - total_assigned_users;
%f = f - f/M * unassigned_users;

end

%% Compute Solution Cost (Fitness)
function f = cost_full(solution)
% computes the cost (fitness) of a given solution
global tetris M T h E R sigma h_sum;

% initialize SINR
SINR = zeros(M,T);

s = solution;

% initialize channel responses and
% interferences for all users
hs = zeros(M,T);
hi = ones(M,T) .* h_sum;

for t = 1:T
```

```
    % initialize active users in current time slot
    active_users = false(1,M);

    % compute channel responses and interference
    % in the t^th time-slot
    for i = 1:s.slots(t).pieces_count
        % for each piece p in the assignment,
        % get the user and assignment
        % value
        p = s.slots(t).pieces(i);
        u = tetris(p).user;
        piece = tetris(p).piece;

        % update active users
        active_users(u) = true;

        % determine active-leds in
        % the piece assignment
        active_leds = find(piece > 0);

        % determine the sum of channel
        % responses of active_leds
        hh = sum(h(u,active_leds));

        % update channel responses
        % and interference values
        hs(u,t) = hs(u,t) + hh;
        hi(u,t) = hi(u,t) - hh;
    end

    % compute SINR for users assigned
    % in the t^th time slots
    for u = 1:M
        if active_users(u)
            SINR(u,t) = E * R^2 * hs(u,t)^2 / ...
            (E * R^2 * hi(u,t)^2 + sigma);
        end
    end
end

% compute data rate per user
DR = sum(log2(1+SINR),2);
% compute the total data rate
f = sum(DR, 'all');
%unassigned_users = M - total_assigned_users;
```

```
%f = f - f/M * unassigned_users;

end


%% Compute Solution Cost (Fitness)
function f = cost(solution)
% computes the cost (fitness) of a given solution
global tetris M T h E R sigma;

% initialize SINR
SINR = zeros(M,T);

s = solution;

assigned_users = false(M,1);
total_assigned_users = 0;

for t = 1:T
    % initialize channel responses and
    % interference in the t^th time-slot
    hs = zeros(M,1);
    hi = zeros(M,1);

    % get assignment in the t^th time slot
    assignment = s.slots(t).assignment;
    n = s.slots(t).pieces_count;
    pieces = s.slots(t).pieces(1:n);

    % determine active-ledes in the assignment
    active_leds = find(assignment > 0);

    % determine users in the assignment
    users = false(1,M);

    % initialize channel interference for
    %the users assigned to active-leds
    %hi(users) = sum(h(users,active_leds),2);

    % compute channel responses and
    %interference in the t^th time-slot
    for p = pieces
        % for each piece p in the assignment,
        % get the user and assignment
        % value
```

```
        user = tetris(p).user;
        piece = tetris(p).piece;

        if ~assigned_users(user)
            assigned_users(user) = true;
            total_assigned_users = total_assigned_users + 1;
        end

        users(user) = true;

        % determine active-leds in the piece assignment
        user_active_leds = find(piece > 0);

        % determine the sum of channel responses of active_leds
        h_sum = sum(h(user,user_active_leds));

        % update channel responses and
        % interference values
        hs(user) = hs(user) + h_sum;
        hi(user) = hi(user) - h_sum;
    end
    active_users = find(users);
    hi(active_users) = hi(active_users) + ...
    sum(h(active_users,active_leds),2);

    % compute SINR for users assigned
    % in the t^th time slots
    for user = active_users
        SINR(user,t) = E * R^2 * hs(user)^2 / ...
        (E * R^2 * hi(user)^2 + sigma);
    end
end

% compute data rate per user
DR = sum(log2(1+SINR),2);
% compute the total data rate
f = sum(DR, 'all');
end

%% Plot Room
function f = plot_room()
% plots LEDs and users locations
global M PD LED L W;
shg
f = 0;
```

```
% show LEDs and users locations on a
% scatter plot
scatter(LED(:,1), LED(:,2));
grid minor;
axis([0 L 0 W])
hold on;
scatter(PD(:,1), PD(:,2));
% label users on the plot
users = (1:M)';
labels = num2str(users,'%d');
text(PD(:,1), PD(:,2), labels, 'horizontal',...
'left', 'vertical','bottom');
drawnow;
end

%% Plot Assignment
function f = plot_solution(solution,...
                CurFitness, BestFitness, Tc)
% show the users' assignment on heat-map plot
% map the solution to an NxT matrix
map = map_solution(solution);
shg
f = 0;
% show the assignment on a heat-map
heatmap(map);
% disply the current+best fitnesses
% and current temperature
s = sprintf('Current: %f, Best: %f, Temp: ...
    %f',CurFitness, BestFitness, Tc);
title(s);
drawnow;
end
%% Map Solution
function map = map_solution(solution)
% maps a solution of a strips-game
% problem to an NxT matrix
global N tetris T;
%T = length(solution);
map = zeros(N,T);
for t = 1:T
    % get the number of pieces in the t^th time-slot
    %n = length(solution(t).pieces);
    n = solution.slots(t).pieces_count;

    % get all pieces in the t^th time-slot
```

```
    pieces = solution.slots(t).pieces(1:n);
    for k = 1:n
        % get index of the k^th piece
        index = pieces(k);
        % get user corresponding to the k^th piece
        user = tetris(index).user;
        % get the piece value
        piece = tetris(index).piece;
        % get LEDs indeces corresponding to
        % the assignment represented by
        % the piece
        %leds = find(piece == 1);
        % assign the LED and time slot
        % to the current user
        map(piece == 1,t) = user;
    end
end
end

%% Compute Tetris Pieces
function pieces = compute_tetris_pieces()
% computes the optimal tetris pieces
% for each user based on the number of
% time slots allocated for each user
global h M N A;

% user index: integer number 1..M
% tetris piece: logical vector size N
% indicates LEDs are assigned to user
pieces = struct('user', {}, 'piece', {});

 m = A;
 m = 1;
% set pieces counter
pieces_cnt = 1;
for user = 1:M
    % find active LEDs with channel responses > 0
    active_leds = find(h(user,:) > 0);
    % find channel responses of LEDs
    % assigned to the current user
    hu = h(user,active_leds);
    % sort channel responses in descending order
    [~,I] = sort(hu, 'descend');
    % sort active LEDs in descending order
    % of their cahnnel responses
```

```
        active_leds = active_leds(I);
        % find number of active LEDs
        n = length(active_leds);

        % generate pieces of individual
        % leds for the current user
        for k = 1:n
            % get indeces of the active LEDs
            led = active_leds(k);
            % compute the piece
            piece = false(1,N);
            piece(led) = true;

            % store m copies of the piece in
            % the tetris pieces matrix
            p = struct('user', user, 'piece', piece);
            start_piece = pieces_cnt;
            end_piece = pieces_cnt + m;
            pieces(start_piece:end_piece) = p;
            pieces_cnt = pieces_cnt + m;
        end

        % generate pieces with 2 or more leds
        % for the current user
        for k = 2:n
            % get indeces of the active LEDs
            leds = active_leds(1:k);
            % compute the piece
            piece = false(1,N);
            piece(leds) = true;

            % store m copies of the piece in
            % the tetris pieces matrix
            p = struct('user', user,'piece', piece);
            start_piece = pieces_cnt;
            end_piece = pieces_cnt + m;
            pieces(start_piece:end_piece) = p;
            pieces_cnt = pieces_cnt + m;
        end
    end
end
```

REFERENCES

[1] L. U. Khan, "Visible light communication: Applications, architecture, standardization and research challenges," *Digital Communications and Networks*, vol. 3, no. 2, pp. 78 – 88, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864816300335

[2] "Cisco visual networking index: Global mobile data traffic forecast update, 2016-2021," [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[3] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[4] H. Haas, C. Chen, and D. O'Brien, "A guide to wireless networking by light," *Progress in Quantum Electronics*, vol. 55, pp. 88 – 111, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0079672717300198

[5] H. H. Dobroslav Tsonev, Stefan Videv, "Light fidelity (li-fi): towards all-optical networking," in *Proc. SPIE*, vol. 9007, 2014, pp. 9007 – 9007 – 10.

[6] A. Jovicic, J. Li, and T. Richardson, "Visible light communication: opportunities, challenges and the path to market," *IEEE Communications Magazine*, vol. 51, no. 12, pp. 26–32, December 2013.

[7] Z. Wang, Q. Wang, W. Huang, and Z. Xu, *Introduction to Visible Light Communications*. IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=8268590

[8] "The ieee 802.15.13 multi-gigabit per second optical wireless communications task group," [Online]. Available: http://http://www.ieee802.org/15/pub/TG13.html.

[9] C. Medina, M. Zambrano Nuñez, and K. Navarro, "Led based visible light communication: Technology, applications and challenges – a survey," vol. 8, pp. 482–495, 09 2015.

[10] J. K. Member, J. M. Kahn, John, and R. Barry, "Wireless infrared communications," in *Proc. Of the IEEE*. Kluwer Academic Publishers, 1994, pp. 265–298.

[11] F. R. Gfeller and U. Bapst, "Wireless in-house data communication via diffuse infrared radiation," *Proceedings of the IEEE*, vol. 67, no. 11, pp. 1474–1486, Nov 1979.

[12] C. Gussen, P. Diniz, M. Campos, W. Martins, F. Costa, and J. Gois, "A survey of underwater wireless communication technologies," *Journal of Communication and Information Systems*, vol. 31, no. 1, Oct. 2016.

[13] Z. Zeng, S. Fu, H. Zhang, Y. Dong, and J. Cheng, "A survey of underwater optical wireless communications," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 204–238, Firstquarter 2017.

[14] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 72–83, Jan 2000.

[15] D. Pompili and I. F. Akyildiz, "Overview of networking protocols for underwater wireless communications," *IEEE Communications Magazine*, vol. 47, no. 1, pp. 97–102, January 2009.

[16] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 4, pp. 23–33, oct 2007.

[17] K. Nakamura, I. Mizukoshi, and M. Hanawa, "Optical wireless transmission of 405 nm, 1.45 gbit/s optical im/dd-ofdm signals through a 4.8 m underwater channel," *Opt. Express*, vol. 23, no. 2, pp. 1558–1566, Jan 2015.

[18] C. Shen, Y. Guo, H. M. Oubei, T. K. Ng, G. Liu, K.-H. Park, K.-T. Ho, M.-S. Alouini, and B. S. Ooi, "20-meter underwater wireless optical communication link with 1.5 gbps data rate," *Opt. Express*, vol. 24, no. 22, pp. 25 502–25 509, Oct 2016.

[19] J. Xu, Y. Song, X. Yu, A. Lin, M. Kong, J. Han, and N. Deng, "Underwater wireless transmission of high-speed qam-ofdm signals using a compact red-light laser," *Opt. Express*, vol. 24, no. 8, pp. 8097–8109, Apr 2016.

[20] M. Kong, W. Lv, T. Ali, R. Sarwar, C. Yu, Y. Qiu, F. Qu, Z. Xu, J. Han, and J. Xu, "10-m 9.51-gb/s rgb laser diodes-based wdm underwater wireless optical communication," *Opt. Express*, vol. 25, no. 17, pp. 20 829–20 834, Aug 2017.

[21] C. Gabriel, M. Khalighi, S. Bourennane, P. Leon, and V. Rigaud, "Channel modeling for underwater optical communication," in *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, Dec 2011, pp. 833–837.

[22] ——, "Monte-carlo-based channel characterization for underwater optical communication systems," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 5, no. 1, pp. 1–12, Jan 2013.

[23] S. Tang, Y. Dong, and X. Zhang, "On path loss of nlos underwater wireless optical communication links," in *2013 MTS/IEEE OCEANS - Bergen*, June 2013, pp. 1–3.

[24] V. Guerra, C. Quintana, J. Rufo, J. Rabadan, and R. Perez-Jimenez, "Parallelization of a monte carlo ray tracing algorithm for channel modelling in underwater wireless optical communications," *Procedia Technology*, vol. 7, pp. 11 – 19, 2013, 3rd Iberoamerican Conference on Electronics Engineering and Computer Science, CIIECC 2013.

[25] V. Guerra, O. El-Asmar, C. Suarez-Rodriguez, R. Perez-Jimenez, and J. M. Luna-Rivera, "Statistical study of the channel parameters in underwater wireless optical links," in *3rd IEEE International Work-Conference on Bioinspired Intelligence*, July 2014, pp. 124–127.

[26] W. Liu, D. Zou, P. Wang, Z. Xu, and L. Yang, "Wavelength dependent channel characterization for underwater optical wireless communications," in *2014 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Aug 2014, pp. 895–899.

[27] Y. Dong, H. Zhang, and X. Zhang, "On impulse response modeling for underwater wireless optical mimo links," in *2014 IEEE/CIC International Conference on Communications in China (ICCC)*, Oct 2014, pp. 151–155.

[28] F. Miramirkhani and M. Uysal, "Visible light communication channel modeling for underwater environments with blocking and shadowing," *IEEE Access*, vol. 6, pp. 1082–1090, 2018.

[29] A. Mora, D. Ganger, G. Wells, J. Zhang, X. Hu, C. Zhou, A. Richa, and C. Youngbull, "Ad-hoc multi-hop underwater optical network for deep ocean monitoring," in *2013 OCEANS - San Diego*, Sept 2013, pp. 1–5.

[30] F. Akhoundi, J. A. Salehi, and A. Tashakori, "Cellular underwater wireless optical cdma network: Performance analysis and implementation concepts," *IEEE Transactions on Communications*, vol. 63, no. 3, pp. 882–891, March 2015.

[31] M. V. Jamali, F. Akhoundi, and J. A. Salehi, "Performance characterization of relay-assisted wireless optical CDMA networks in turbulent underwater channel," *CoRR*, vol. abs/1508.04030, 2015.

[32] M. V. Jamali, A. Chizari, and J. A. Salehi, "Performance analysis of multi-hop underwater wireless optical communication systems," *IEEE Photonics Technology Letters*, vol. 29, no. 5, pp. 462–465, March 2017.

[33] A. Tabeshnezhad and M. A. Pourmina, "Outage analysis of relay-assisted underwater wireless optical communication systems," *Optics Communications*, vol. 405, pp. 297 – 305, 2017.

[34] X. Ma, F. Yang, S. Liu, and J. Song, "Channel estimation for wideband underwater visible light communication: a compressive sensing perspective," *Opt. Express*, vol. 26, no. 1, pp. 311–321, Jan 2018.

[35] C. D. Mobley, B. Gentili, H. R. Gordon, Z. Jin, G. W. Kattawar, A. Morel, P. Reinersman, K. Stamnes, and R. H. Stavn, "Comparison of numerical models for computing underwater light fields," *Appl. Opt.*, vol. 32, no. 36, pp. 7484–7504, Dec 1993.

[36] J. Poliak, P. Pezzei, E. Leitgeb, and O. Wilfert, "Link budget for high-speed short-distance wireless optical link," in *2012 8th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*, July 2012, pp. 1–6.

[37] F. . U. M. Elamassie, Mohammed Miramirkhani, "Performance characterization of underwater visible light communication," *IEEE Transactions on Communications.*, 2018.

[38] M. Khalighi, T. Hamza, S. Bourennane, P. Léon, and J. Opderbecke, "Underwater wireless optical communications using silicon photo-multipliers," *IEEE Photonics Journal*, vol. 9, no. 4, pp. 1–10, Aug 2017.

[39] M. Kong, Y. Chen, R. Sarwar, B. Sun, Z. Xu, J. Han, J. Chen, H. Qin, and J. Xu, "Underwater wireless optical communication using an arrayed transmitter/receiver and optical superimposition-based pam-4 signal," *Opt. Express*, vol. 26, no. 3, pp. 3087–3097, Feb 2018.

[40] W. Liu, Z. Xu, and L. Yang, "Simo detection schemes for underwater optical wireless communication under turbulence," *Photon. Res.*, vol. 3, no. 3, pp. 48–53, Jun 2015.

[41] F. Ahmed, S. Ali, and M. Jawaid, "A review of modulation schemes for visible light communication," vol. 18, pp. 117–125, 02 2018.

[42] Y. Zhang, Z. Zhang, Z. Huang, H. Cai, L. Xia, and J. Zhao, "Apparent brightness of leds under different dimming methods," vol. 6841, 01 2008.

[43] S. Rajagopal, R. D. Roberts, and S. Lim, "Ieee 802.15.7 visible light communication: modulation schemes and dimming support," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 72–82, March 2012.

[44] H. Elgala, R. Mesleh, and H. Haas, "Indoor optical wireless communication: potential and state-of-the-art," *IEEE Communications Magazine*, vol. 49, no. 9, pp. 56–62, September 2011.

[45] G. Miao, J. Zander, K. W. Sung, and S. Ben Slimane, *Fundamentals of Mobile Data Networks*. Cambridge University Press, 2016.

[46] A. M. Vegni and T. D. C. Little, "Handover in vlc systems with cooperating mobile devices," in *2012 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2012, pp. 126–130.

[47] T. Nguyen, M. Z. Chowdhury, and Y. M. Jang, "A novel link switching scheme using pre-scanning and rss prediction in visible light communication networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 293, Dec 2013.

[48] X. Bao, X. Zhu, T. Song, and Y. Ou, "Protocol design and capacity analysis in hybrid network of visible light communication and ofdma systems," vol. 63, pp. 1770–1778, 05 2014.

[49] E. Dinc, O. Ergul, and O. B. Akan, "Soft handover in ofdma based visible light communication networks," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Sept 2015, pp. 1–5.

[50] M. S. Demir, S. M. Sait, and M. Uysal, "Unified resource allocation and mobility management technique using particle swarm optimization for vlc networks," *IEEE Photonics Journal*, pp. 1–1, 2018.

[51] X. Li, R. Zhang, and L. Hanzo, "Cooperative load balancing in hybrid visible light communications and wifi," *IEEE Transactions on Communications*, vol. 63, no. 4, pp. 1319–1329, April 2015.

[52] L. Li, Y. Zhang, B. Fan, and H. Tian, "Mobility-aware load balancing scheme in hybrid vlc-lte networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2276–2279, Nov 2016.

[53] Y. Wang and H. Haas, "Dynamic load balancing with handover in hybrid li-fi and wi-fi networks," *Journal of Lightwave Technology*, vol. 33, no. 22, pp. 4671–4682, Nov 2015.

[54] Y. Wang, X. Wu, and H. Haas, "Load balancing game with shadowing effect for indoor hybrid lifi/rf networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2366–2378, April 2017.

[55] M. Obeed, A. M. Salhab, S. A. Zummo, and M. Alouini, "Joint optimization of power allocation and load balancing for hybrid vlc/rf networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 5, pp. 553–562, May 2018.

[56] B. Ghimire and H. Haas, "Resource allocation in optical wireless networks," in *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*, Sept 2011, pp. 1061–1065.

[57] R. K. Mondal, M. Z. Chowdhury, N. Saha, and Y. M. Jang, "Interference-aware optical resource allocation in visible light communication," in *2012 International Conference on ICT Convergence (ICTC)*, Oct 2012, pp. 155–158.

[58] O. Babatundi, L. Qian, and J. Cheng, "Downlink scheduling in visible light communications," in *2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP)*, Oct 2014, pp. 1–6.

[59] Y. Tao, X. Liang, J. Wang, and C. Zhao, "Scheduling for indoor visible light communication based on graph theory," *Opt. Express*, vol. 23, no. 3, pp. 2737–2752, Feb 2015. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-23-3-2737

[60] F. Jin, X. Li, R. Zhang, C. Dong, and L. Hanzo, "Resource allocation under delay-guarantee constraints for visible-light communication," *IEEE Access*, vol. 4, pp. 7301–7312, 2016.

[61] X. Wu, M. Safari, and H. Haas, "Bidirectional allocation game in visible light communications," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–5.

[62] D. Nguyen, N. Nguyen, N. Nguyen, and K. Spirinmanwat, "Light beam allocation algorithm for eliminating interference in visible light communications," in *2016 International Conference on Advanced Technologies for Communications (ATC)*, Oct 2016, pp. 419–424.

[63] W. Wu, F. Zhou, and Q. Yang, "Dynamic network resource optimization in hybrid vlc and radio frequency networks," in *2017 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, May 2017, pp. 1–7.

[64] F. Yunlong, L. Jianhui, L. Qing, and Z. Xiaoyi, "Downlink channel allocation of visible light communication network based on graph coloring and traffic fairness," *Procedia Computer Science*, vol. 107, pp. 667 – 673, 2017, advances in Information and Communication Technology: Proceedings of 7th International Congress of Information and Communication Technology (ICICT2017). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917304210

[65] F. Seguel, A. D. Firoozabadi, P. Adasme, I. Soto, N. Krommenacker, and C. Azurdia-Meza, "A novel strategy for led re-utilization for visible light communications," *Optik*, vol. 151, pp. 88 – 97, 2017, optical Wireless Communication Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0030402617313347

[66] R. Jiang, Z. Wang, X. Zhu, and Q. Wang, "Interference-free led allocation for visible light communications with fisheye lens," *Journal of Lightwave Technology*, vol. 36, no. 3, pp. 626–636, Feb 2018.

[67] R. C. Kizilirmak, O. Narmanlioglu, and M. Uysal, "Centralized light access network (c-lian): A novel paradigm for next generation indoor vlc networks," *IEEE Access*, vol. 5, pp. 19 703–19 710, 2017.

[68] Y. S. Eroğlu, . Güvenç, A. Şahin, Y. Yapıcı, N. Pala, and M. Yüksel, "Multi-element vlc networks: Led assignment, power control, and optimum combining," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 1, pp. 121–135, Jan 2018.

[69] E. D. Demaine, S. Hohenberger, and D. Liben-Nowell, "Tetris is hard, even to approximate," *CoRR*, vol. cs.CC/0210020, 2002. [Online]. Available: http://arxiv.org/abs/cs.CC/0210020

[70] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[71] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. P45, no. 1, pp. 41–51, Jan 1985.

[72] S. M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*, 1st ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999.

VITAE

- Name: Khaled Abdul-Aziz Eid Al-Utaibi
- Nationality: Saudi
- Date of Birth: 10/01/1393
- Email: *alutaibi@uoh.edu.sa*
- Permenant Address: Hail, Saudi Arabia

- Master of Science, Computer Engineering, 2002, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
- Bachelor of Science, Computer Engineering, 1997, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia