M.Sc. Engg. Thesis

Intelligent Dynamic Spectrum Access Exploiting A Synergy Between Genetic Algorithm And Local Search

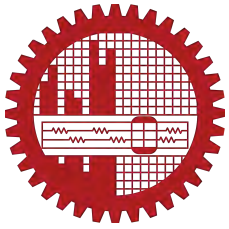
by

Md. Jahidul Islam

Submitted to

Department of Computer Science and Engineering

in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering



Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology (BUET)

Dhaka 1000

February 2015

Dedicated to my loving parents

Author's Contact

Md. Jahidul Islam

Lecturer,

Department of Computer Science & Engineering,

United International University (UIU), Dhaka.

Email: jahid@cse.uiu.ac.bd, jahidul@csebuet.org

The thesis titled "Intelligent Dynamic Spectrum Access Exploiting A Synergy Between Genetic Algorithm And Local Search", submitted by Md. Jahidul Islam, Roll No. **0412052066P**, Session April 2012, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on February 4, 2015.

Board of Examiners

1. _____

Dr. Md. Monirul Islam — Chairman
Professor — (Supervisor)
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

2. _____

Dr. A. B. M. Alim Al Islam — Member
Assistant Professor — (Co-Supervisor)
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

3. _____

Dr. Mohammad Mahfuzul Islam — Member
Head and Professor — (Ex-Officio)
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

4. _____

Dr. Md. Shohrab Hossain — Member
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

5. _____

Dr. Chowdhury Mofizur Rahman — Member
Professor — (External)
Department of Computer Science and Engineering
United International University, Dhaka.

Candidate's Declaration

This is hereby declared that the work titled "Intelligent Dynamic Spectrum Access Exploiting A Synergy Between Genetic Algorithm And Local Search", is the outcome of research carried out by me under the supervision of Dr. Md. Monirul Islam and co-supervision of Dr. A. B. M. Alim Al Islam, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Md. Jahidul Islam

Candidate

Acknowledgment

Foremost, I express my heart-felt gratitude to my supervisor, Dr. Md. Monirul Islam, and co-supervisor, Dr. A. B. M. Alim Al Islam, for their constant supervision of this work. They helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. Their patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I would also want to thank the members of my thesis committee: Dr. Mohammad Mahfuzul Islam, Dr. Md. Shohrab Hossain, and specially the external member Dr. Chowdhury Mofizur Rahman, for their encouragements, insightful comments, and valuable suggestions.

I am also thankful to Chowdhury Sayeed Hyder (PhD Candidate, Michigan State University, USA), Sukarna Barua (Assistant Professor, CSE-BUET), and Tanvir Ahmed Khan (Lecturer, CSE-BUET). I sought help from them a number of occasions regarding simulation set-up and performance evaluation of this thesis. In addition, I am grateful to Dr. Swakkhar Swatabta (Assistant Professor, CSE-UIU), Novia Nurain (Assistant Professor, CSE-UIU), Sajjadur Rahman (Lecturer, CSE-BUET), and Himel Dev (Lecturer, CSE-BUET) for their help and valuable suggestions regarding the writing and presentation of this thesis.

Last but not the least, I remain ever grateful to my beloved parents, who always exists as sources of inspiration behind every success of mine.

Abstract

This thesis presents a novel hybrid dynamic spectrum access technique for multi-channel single-radio cognitive radio networks. Existing classical and stochastic approaches exhibit different advantages and disadvantages depending on network topology and architecture. Our proposed approach exploits a delicate balance between these two types of approaches for extracting advantages from both of them while limiting their disadvantages. We exploit a synergy between genetic algorithm-based stochastic search and classical local search to design a highly scalable and efficient dynamic spectrum access technique. Additionally, we boost up the performance of our algorithm through designing new genetic operators.

Besides, proper and thorough performance evaluation of existing approaches using a discrete event simulator is yet to be performed in the literature. To address this issue, we simulate several existing approaches using a widely used discrete event simulator called `ns-2`. We evaluate the performance of our proposed technique in `ns-2` on the basis of various standard performance metrics. In the evaluation, we compare the performance of our proposed technique with that of the state-of-the-art approaches. Simulation results demonstrate significant performance improvement using our proposed approach over the existing ones.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Radio spectrum is a natural, however limited, resource regulated by governmental or international agencies. The spectrum is assigned to license holders known as *Primary Users (PUs)* on a long-term basis using a fixed spectrum assignment policy [8]. However, it has been reported that, a large portion of the assigned spectrum remains under-utilized even in recent times [1], as depicted in Figure 1.1. This figure shows that some spectrum bands are heavily used, whereas few other bands are sparsely used, leading to an inconsistent and inefficient spectrum utilization.

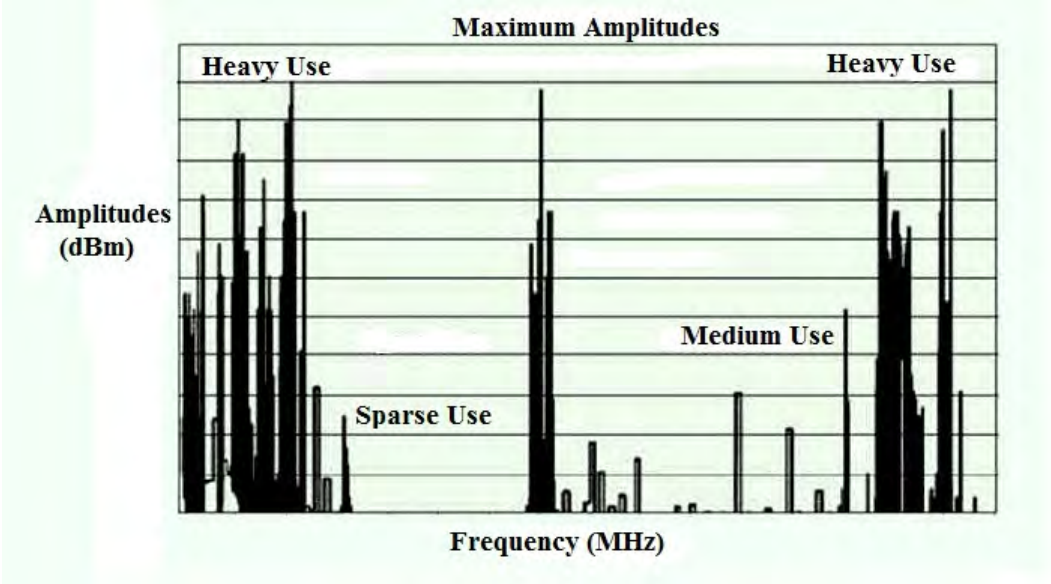


Figure 1.1: Inefficient spectrum utilization (source: [1])

We can visualise this inefficient spectrum utilization with a practical example. Let us consider a

road, which is used by different types of users for transportation. There are separate and dedicated lanes for each types of users. As depicted in Figure 1.2, the first lane is dedicated for cell-phone users and the second one for television users. The third lane is reserved for 'royal' users (may be for the king and his family!), fourth one for emergency use (may be for doctors and armed forces!), and the last lane is reserved for satellite users.

Now, obviously the first lane will always be crowded due to the large number of cell-phone users, whereas, the third and fourth lane is going to be vacant most of the time. Therefore, we will have severe congestion in a particular lane and very little utilization of the other lanes. The fixed user-specific lanes or boundaries are responsible for such an inefficient *road utilization*[1].

This is exactly what happens in the fixed spectrum access policy. Few frequency bands such as the mobile frequencies are getting crowded where other bands such as television and satellite frequencies are underutilized. How to solve this problem? Well, at first, lets solve the inefficient *road utilization* problem.

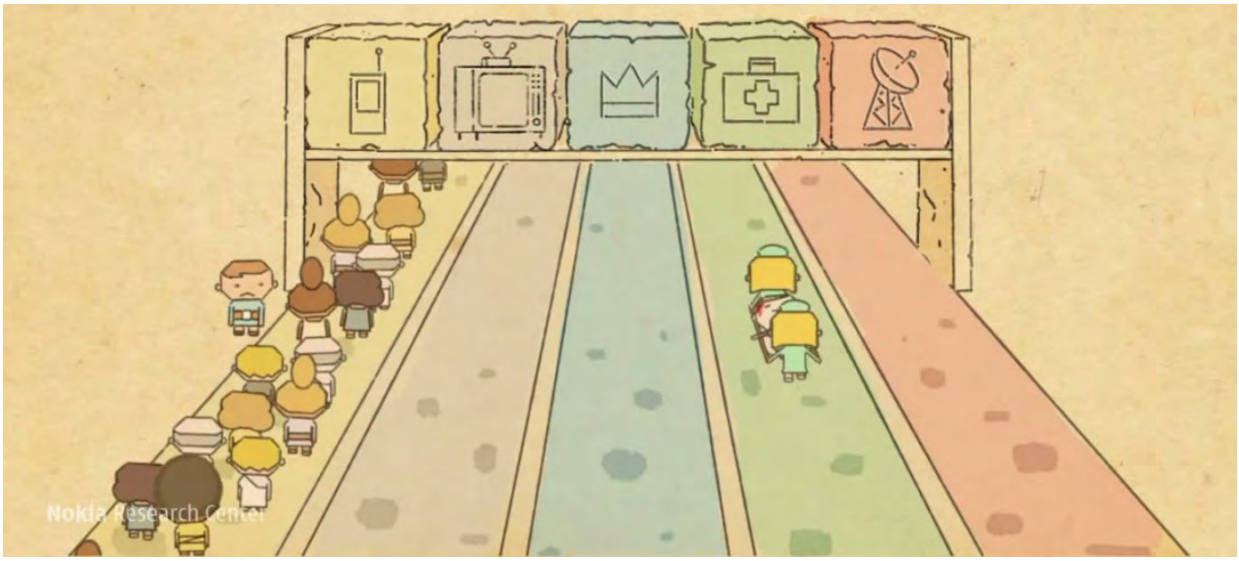


Figure 1.2: Inefficient 'road utilization' with fixed lane access

We can achieve a much better road-utilization if we remove the lane boundaries allowing all types of users to use any vacant lane (as shown in Figure 1.3). That is, users may use any lane at any point of time if it is vacant. In case of congestion in a lane, they are allowed to switch to another vacant lane. This *opportunistic lane access* will result in an efficient road utilization. However, we need to ensure no congestion to the emergency and high-priority users. To do so, we also need to be

[1]This example is originally provided in a video made by Nokia Research Center [9]

able to *reserve* a particular lane temporarily for special users (as shown in Figure 1.4). Therefore, to enable this opportunistic lane access, the users need to be able to sense and switch lanes dynamically. Besides, we need some kind of mechanism to reserve a particular lane for emergency users.

Figure 1.3: Opportunistic lane access with no user-specific boundaries

The solution of the inefficient *spectrum utilization* problem is somewhat similar. We allow the unlicensed users, known as *Secondary Users (SUs)*, to opportunistically use the unused portions of the wireless spectrum, without interfering the Primary Users (PUs) ([10], [11]). That is, we enable opportunistic spectrum access among the users just like the opportunistic lane access mechanism. Therefore, in this case, the SUs need to be able to *sense* the radio environment and switch spectrum bands in case of congestion. Besides, we need to ensure no interference to PUs by reserving their spectrum bands whenever they are in transmission. This whole process is termed as *dynamic spectrum management*.

Cognitive Radio (CR) technology ([2], [3], [7]) enables such opportunistic spectrum access, which results in more efficient spectrum utilization. CRs are aware of their surroundings and bandwidth availability and are able to dynamically tune the spectrum usage based on location, nearby radios, time of day and other factors. This provides for a more efficient use of the spectrum as well as reducing power consumption, and enabling high priority communications to take precedence if needed [9].

CRs are mostly based on Software Defined Radio (SDR) [12]. SDRs add programmability to radio

Figure 1.4: Dynamic lane reservation for high-priority users

devices, increasing their flexibility to operate on different spectrum bands and with different modulations. An SDR transceiver is able to adapt its transmission parameters to the radio environment, which can vary over time. This ability allows users to access any portion of the free spectrum and not just a specific spectrum band, which is the case in current radios (i.e. 3G, 802.11, GSM, etc). Networks that are proposed to use CRs are called Cognitive Radio networks (CRNs).

A simple network architecture of CRNs is provided in Figure 1.5, which shows coexistence of secondary networks (for SUs) and primary networks (for PUs). These PUs and SUs share a set of spectrum bands for their transmissions where SUs are only allowed to use a particular band if it does not interfere any PU transmissions. Such opportunistic spectrum access is termed as Dynamic Spectrum Access (DSA).

DSA ([3], [7]) provides the basic functionality of Cognitive Radio networks (CRNs) through opportunistically assigning the most appropriate spectrum fragment to CR devices while ensuring avoidance of interference to PUs operating in the same geographical area. An overview of DSA is provided in Figure 1.6 through a time versus frequency illustration for a particular SU. Here, the shadowed regions represent the spectrum fragments that are being used by PUs, whereas, the unshaded regions represent the unused spectrum bands for a given time interval. As demonstrated by this figure, the SU uses an unused spectrum fragment (i.e., spectrum hole) at any particular time interval. While accessing, if that particular spectrum fragment becomes unavailable (i.e., accessed by a PU), the SU

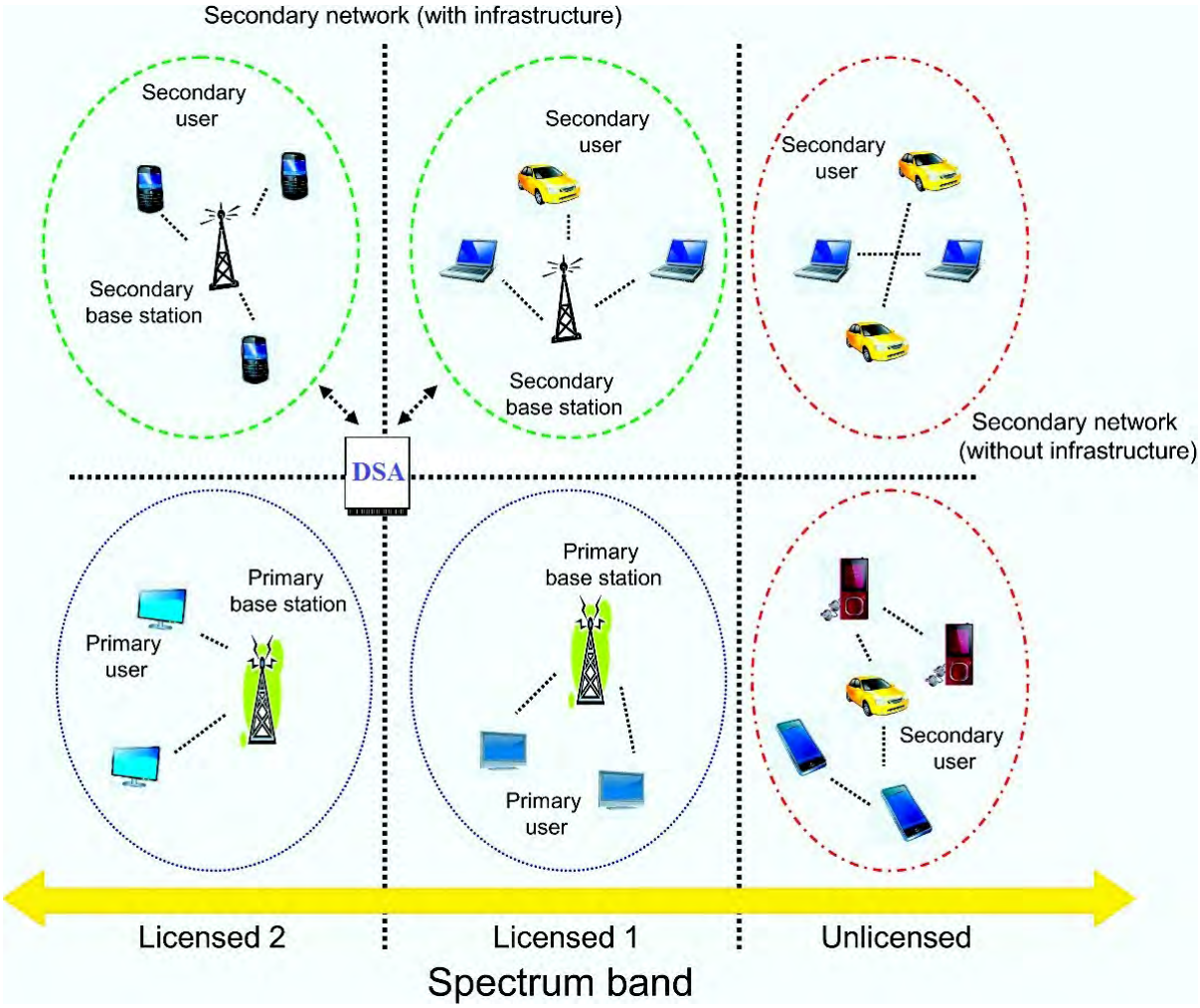


Figure 1.5: Cognitive Radio Network (CRN) architecture (source: [2])

switches to another unused spectrum fragment. This opportunistic spectrum access is called DSA. Mechanism of DSA, shown in Figure 1.7, consist of four basic functionalities [3]:

i. **Spectrum Sensing:** Identification of unused spectrum fragments (also termed as *spectrum holes*),

ii. **Spectrum Decision:** Selection of the best available spectrum fragment according to some criteria [7] (such as minimizing interference, increasing network throughput, etc.),

iii. **Spectrum Mobility:** Vacating the spectrum fragment when a PU within the same region wants to access the same fragment, and

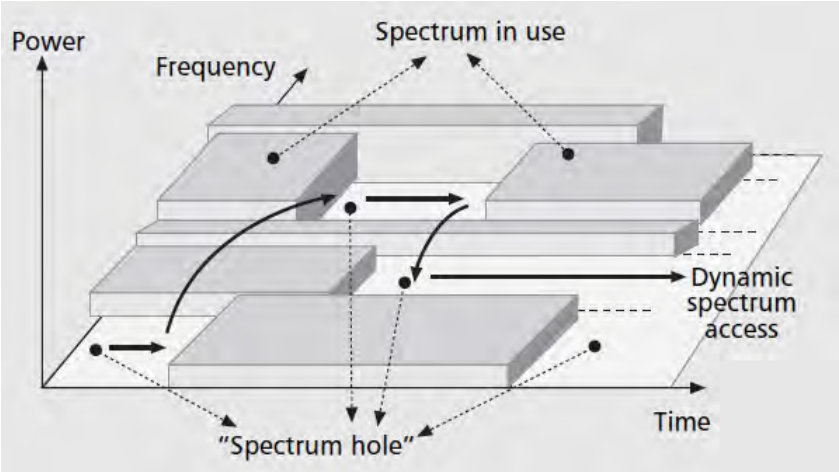


Figure 1.6: Dynamic Spectrum Access (DSA) (source: [3], [4])

iv. **Spectrum Sharing:** Coordinating access to the spectrum fragment being used with other SUs.

When an SU needs to access a spectrum fragment for data transmission, at first, we find available portions of the spectrum by sensing the medium. That is, we avoid spectrum fragments that are currently being used by the PUs. Then, DSA algorithm makes the spectrum assignment decision according to some criteria [7], such as minimizing interference, increasing network throughput, ensuring network fairness, etc. When a PU is detected accessing a spectrum fragment that is currently being used by a SU within its transmission range, the SU stops its current transmission, leaves that spectrum fragment, and requests for other available spectrum fragment. In addition to avoiding interference to PUs, SUs coordinate access to the spectrum fragment being used with other SUs to ensure efficient spectrum utilization and better network performance.

In this thesis, we design a novel hybrid DSA technique for multi-channel single-radio CRNs. Designing efficient DSA techniques for CRNs has been a well-researched topic in recent years. In the following sections, we discuss the motivations and contributions of our work in this affair.

1.1 Motivations of Our Work

It may seem a straight forward problem that we just need to sense the medium and assign the unused spectrum fragments to the SUs. However, due to the presence of different network architectures and varied network topologies, the task of DSA becomes rather complex. The basic constraint of DSA algorithms is to ensure efficient spectrum utilization of SUs without interfering PUs. The most

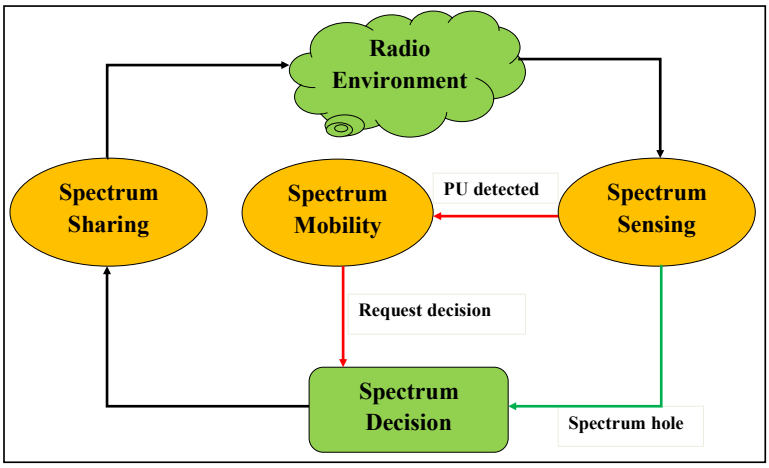


Figure 1.7: Mechanism of DSA

challenging part of a DSA algorithm is to maximize the network performance subject to this basic constraint. Network performance is measured in terms of several QoS parameters [7] such as throughput, delay, fairness, energy efficiency, network connectivity, etc. The overall network performance of CRNs depends largely on optimum spectrum access ([7]-[11]) of the SUs. Therefore, it is very important, however extremely challenging, to design an efficient DSA technique to ensure optimum spectrum access for the SUs over varied network scenarios.

There are a number of research studies that demonstrate various DSA techniques. Graph-theory based algorithms ([2], [7], [13]-[19]) are the most commonly used classical approaches. These approaches visualize the network as a graph where the vertices correspond to CR users and edges correspond to connections among them. Such approaches generally construct *network conflict graph* [7] to capture interference between neighbouring SUs ([13], [14], [20]). These approaches also use graph coloring ([15], [18], [21]), where the DSA problem is mapped into a graph coloring problem. In addition, approaches based on bipartite graphs and layered graphs [7] are also common for DSA. However, most of these approaches consider networks having only SUs [7]. Such sole consideration of SUs, ignoring the presence of PUs, are not suitable for pragmatic deployment of CRNs.

Many studies on DSA adopt game theory based approaches ([22]-[27]) as the concept of game theory fits quite well with the DSA problem. There are two types of games, cooperative and non-cooperative, based on whether the players (i.e., CR users) exchange information regarding their decisions or not. Most DSA algorithms based on game theory formulate a game and try to find the optimal solution through Nash equilibrium. Auction theory [24] based approaches ([22], [25]) also fall

into this category of DSA algorithms. In such approaches, SUs contend for a set of channels and a regulator conducts an auction to sell rights on accessing these channels. The problem of game theory based approaches is that the utility function and game formulation used in these approaches must be very carefully structured to achieve equilibrium, which is not always guaranteed in reality.

In addition to graph theory and game theory based approaches, agent based learning ([28], [29]), linear programming [30], and fuzzy logic based approaches [31] are also proposed in recent studies. Agent based learning approaches demand problem specific design, while the approaches based on linear programming adopt few assumptions, which are not always valid in reality [7]. Besides, a fuzzy system is not scalable as a large number of rules are required for performing DSA considering all different parameters that can affect DSA decision [7].

As the DSA problem belongs to the class of NP-complete problems [32], all the approaches mentioned above demand high computational overhead and often lead to starvation with an increasing number of CR users. To overcome these issues, stochastic search methods such as heuristic search algorithms ([18], [21], [32], [33]) and evolutionary algorithms ([34]-[37]) are proposed for DSA in the literature. Such approaches are less sensitive to variations in problem characteristics and dimensionality. However, the disadvantage of these approaches is that, they often get stuck in locally optimal solutions, which can be far from the globally optimal solution. Consequently, the two classes of approaches for DSA, i.e., classical approaches and stochastic approaches, exhibit different advantages and disadvantages (summarised in Table 1.1). Additionally, although using the notion of classical local search with stochastic methods facilitate better avoidance of locally optimal solutions and faster convergence in many optimization problems [38], such hybrid approaches are yet to be studied for DSA in CRNs. Moreover, even though there are several approaches proposed for DSA, proper and thorough performance evaluation of these approaches using a discrete event simulator is yet to be performed in the literature, as they are mostly evaluated through numerical simulation.

In this work, we address all the issues mentioned above. We devise an intelligent DSA algorithm having properties of both classical and stochastic methods. Afterwards, we evaluate its performance and that of other state-of-the-art algorithms in `ns-2`. Our proposed approach exploits a synergy between a Genetic Algorithm (GA) based stochastic method and classical local search based novel genetic operator. Here, at first, we perform an empirical study on the performance of GA in DSA with basic genetic operators that are most commonly used in the literature. Subsequently, we devise two novel genetic operators, which are neighborhood-based crossover and local search based survivor

Table 1.1: Pros and cons of the state-of-art DSA techniques in CRNs

Issues	**Classical approaches**	**Stochastic approaches**
Scalability	Limited	Ok
Ease of Formulation	Limited	Ok
Robustness	Limited	Ok
Consideration of PUs	Not always	Ok
Convergence	Ok	Not always
Problem Specific Operator/Parameter Design	Ok	Not always

selection. Here, our motivation is to use efficient searching of GA in addition to avoiding local optima solutions through Local Search (LS). Next, we present the details of our contributions.

1.2 Our Contributions

We name our proposed DSA approach as GALS. The hybrid nature of GALS facilitates establishing a delicate balance between exploration and exploitation, which makes it more efficient and more capable in ensuring faster convergence. To the best of our knowledge, we are the first to propose this type of hybrid approach for DSA. Consequently, we make the following contributions in this paper:

- We perform a thorough performance evaluation of the basic genetic operators and parameter values using discreet event simulator. The empirical study presents the combination of genetic operators and parameter values that performs the best with GA based DSA.

- Subsequently, we design novel hybrid genetic operators, which demonstrate significantly better performance than the previously found best combination of genetic operators. These novel and efficient genetic operators, along with the tuned parameter values, result in a highly scalable and efficient DSA technique (i.e., GALS) for multi-channel, single-radio CRNs.

- We evaluate the performance of GALS using a discrete event simulator called *Cognitive Radio Cognitive Network (CRCN) simulator* [39], which is based on ns-2. We perform necessary modifications in the basic CRCN simulator to enable the spectrum sharing and spectrum mobility features in our evaluation. The modified simulator can be exploited for future research purpose in evaluating performance of DSA algorithms in CRNs.

- Using the CRCN simulator, we compare the performance of GALS with the most widely used

state-of-the-art DSA algorithms such as DSA approaches based on graph-theory, game-theory, heuristics, and evolutionary algorithms. Simulation results suggest significant performance improvement using GALS compared to these state-of-the-art algorithms. In particular, GALS demonstrates remarkable performance improvement on fairness of the network, which is considered as the most challenging performance metric to be improved in distributed CRNs ([7], [40]).

1.3 Outline of Our Thesis

This is how the rest of this book is organized. In Chapter 2, we elaborate on the background of the DSA problem in CRNs and analyse this problem from different viewpoints highlighting related works in each cases. Then, in Chapter 3, we present the network model and summarise the modifications that we performed in the basic CRCN simulator. Next to that, we investigate the performance of GA-based DSA with basic genetic operators in Chapter 4. On the basis of results found from this chapter, we devise novel genetic operators that we use in GALS. We present these novel genetic operators in Chapter 5. Then, we illustrate the detailed GALS algorithm and compute its computational complexity in Chapter 6. Finally, in Chapter 7, we present a comprehensive analysis of the performance of GALS algorithm in terms various QoS parameters. This chapter also include the performance comparison of GALS with other state-of-the-art algorithms and findings of our simulation results. At last, we draw our conclusions and present possible future studies in Chapter 8.

Chapter 2

Background and Related Work

The fixed spectrum assignment policies of governmental agencies result in wastage of valuable wireless spectrum resources. Cognitive Radio (CR) is a promising technology to ensure efficient spectrum utilization by exploiting those unused portions of the spectrum. CR is a radio that can sense the environment dynamically and adjust its radio operating parameters accordingly [3]. The Federal Communications Commission (FCC) proposed [41] the following term for CR in 2003:

> *CR is a radio that can change its transmitter parameters based on its interaction with the environment in which it operates. This interaction may involve active negotiation or communications with other spectrum users and/or passive sensing and decision making within the radio.*

Wireless networks that aims to use the CR technology, are called Cognitive Radio Networks (CRNs). A possible CRN architecture is depicted in Figure 2.1. Due to the presence of two types of users (i.e., PUs and SUs), two types of network co-exist in CRNs. Primary Networks (PNs) are the existing wireless network infrastructures, such as GSM, UMTS, TV broadcast etc., that have been assigned licenses to operate in specific frequency bands. These networks consist of primary base stations and PUs. Primary base stations are used in infrastructure mode of wireless networks and hold a spectrum license for communicating with the PUs. Generally, the primary base stations do not have any functionalities for sharing the spectrum with the SUs [7].

On the other hand, Secondary Network (SNs) are networks whose users (i.e., SUs) do not have license to access any frequency bands and use CR technology to temporarily access the spectrum in an opportunistic manner [3]. SNs can operate in either centralised mode (with infrastructure) or

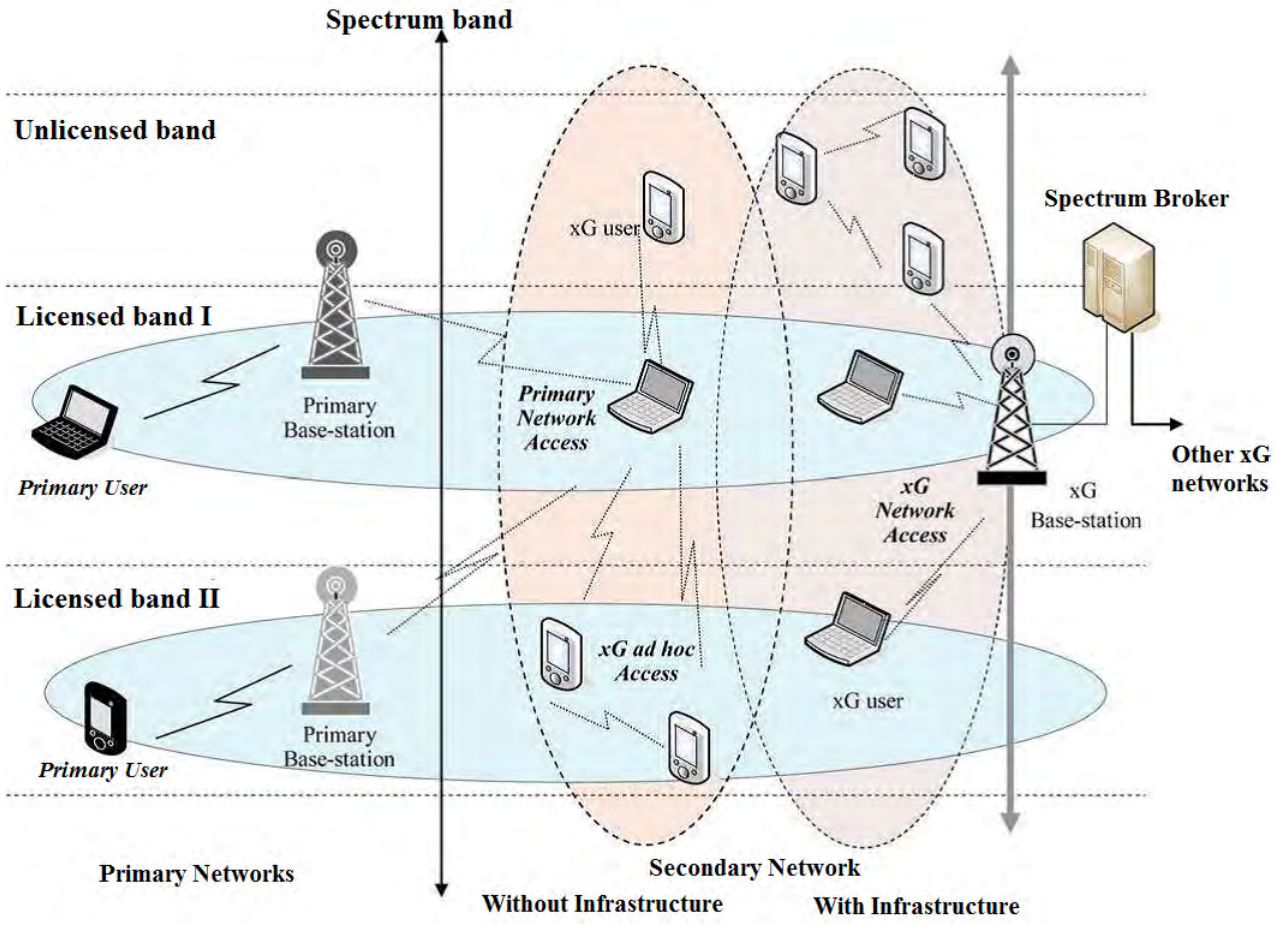
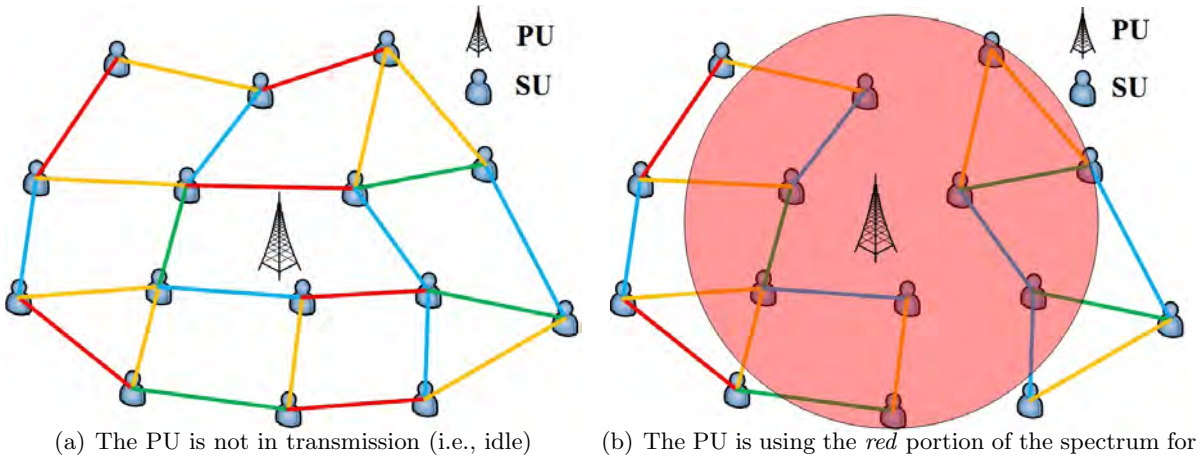


Figure 2.1: CRN architecture (source: [5])

distributive mode (without infrastructure). In centralised mode, a secondary base station or a central entity (spectrum broker) coordinates spectrum usage among SUs. On the contrary, SUs take decisions either by themselves or by cooperating with their neighbours in distributive mode.

CR technology allows SUs to sense the spectrum for unused portions and use the most suitable ones, according to some pre-defined criteria, through Dynamic Spectrum Access (DSA). DSA is the key mechanism that limits the interference between CR users, enabling a more efficient usage of the wireless spectrum. We already discussed the detailed mechanism of DSA in the previous chapter. Now, in Figure 2.2, we illustrate how DSA enables SUs to access spectrum holes without interfering PUs. Here, we represent several possible SU connections with coloured lines between them. Each different color corresponds to a different spectrum fragment. In Figure 2.2(a), the PU is not in transmission (i.e., idle). Therefore, all the spectrum fragments are available to the SUs. On the other hand, in Figure 2.2(b), the PU is active and using the red spectrum fragment. As a result, no SUs

within this PU's transmission range are now allowed to use this spectrum fragment.



(a) The PU is not in transmission (i.e., idle)

(b) The PU is using the *red* portion of the spectrum for transmission

Figure 2.2: A CRN scenario with DSA operation. Each coloured line between SUs corresponds to possible connections between them, where different color represents different spectrum fragments. (source: [6])

Because CRs are able to sense, detect, and monitor the surrounding RF environment such as interference and access availability, and reconfigure their own operating characteristics to best match outside situations, cognitive communications can increase spectrum efficiency and support higher bandwidth service. As a result, CRs can be employed in many applications [2]. CRs can be used in military communications, as currently the capacity of military communications is limited by radio spectrum scarcity. Therefore, CRNs can liberate these limitations in addition to ensuring adaptive, seamless, and secure military communications [2].

CRNs can also be implemented to enhance public safety and homeland security. A natural disaster or terrorist attack can destroy existing communication infrastructure, so an emergency network becomes indispensable to aid the search and rescue. CRNs can be very effective in such cases. Another very promising application of CR is in the commercial markets for wireless technologies. Since CR can intelligently determine which communication channels are in use and automatically switches to an unoccupied channel, it provides additional bandwidth and versatility for rapidly growing data applications.

It is evident from the above applications of CRNs that DSA is the main feature of CRNs. The performance of DSA determines the overall effectiveness of CRNs. Now, we demonstrate state-of-the-art techniques that are used for CRNs. There are a number of DSA techniques proposed in the

literature. We present them according to architectural, spectrum sharing, and algorithmic point of view.

2.1 Architectural Viewpoint

As per our discussion above, CRNs can operate in either centralized or distributed mode. Consequently, DSA approaches can also be classified in two categories [7].

2.1.1 Centralized DSA

Centralized DSA approaches ([40], [42]) require a central entity to decide on assigning channels to cognitive nodes. This central entity may be a separate node called *spectrum server* or *spectrum broker* (Figure 2.1), or a central base station that collects spectrum and radio information from all SUs either periodically or on-demand.

In centralized DSA techniques, it is easier to maximize the overall network throughput and to minimize interference between SUs and in general the network performance. Besides, the spectrum server can also be used to achieve fairness in terms of either allocated spectrum or throughput minimizing the number of greedy users that use many spectrum bands to increase their throughput, causing problems to other users. Connectivity maintenance is another key advantage because the global view of the network can help avoid disconnections.

Moreover, the spectrum server can use priorities to links or nodes with constrained interfaces to ensure that these links will have high throughput, i.e. for links close to gateways. On the other hand, a major disadvantage of centralized cognitive DSA is that it induces signalling overhead in the network, because of the need to exchange measurements between the SUs and the spectrum server. In addition, if the spectrum server fails due to crashes or power failures, then spectrum assignment will not be possible and each SU will choose its own channel(s) independently, leading to contention and unfairness.

2.1.2 Distributed DSA

In distributed DSA ([13], [43], [44]), no central entity is responsible for assigning channels to cognitive users. In this case, users take decisions either by themselves or by cooperating with their neighbours.

In such approaches, each node selects the spectrum fragment with the minimum traffic load or the one that creates minimum interference to its neighbors (or according to some other metric).

Distributed DSA is usually more flexible, as it can quickly adapt to possible changes or network outages in a localized manner. This is a much faster process compared to a centralized one. Another advantage is that it incurs a lower signalling overload in the network, since only neighbour nodes have to exchange messages. However, fairness can only be achieved locally for a group of neighbour SUs, which can be far away from global fairness over the whole network. Another issue with distributed schemes is that the decisions are based on the exchange of measurements between the SUs. Therefore, missing or inaccurate information can significantly affect the decision. Distributed DSA can usually take adequate decisions in cases of low traffic load. However, in high traffic load situations, a centralized scheme, having knowledge of the traffic in the whole network, can take better decisions.

Although distributed architectures have more operational challenges than centralized ones, distributed CRNs are significantly more scalable and resilient. It is almost impossible to design a centralised network without sacrificing scalability and causing high signalling overhead. Therefore, distributed architecture is the preferred choice for pragmatic deployment of scalable CRNs.

2.2 Spectrum Sharing Viewpoint

Considering access technology and sharing methodology, we can classify spectrum sharing into several categories. We discuss this categories in comparative fashion in the following subsections.

2.2.1 Overlay Sharing vs Underlay Sharing

Considering the access technology of the SUs, spectrum sharing can be divided in two categories ([2], [45]): *overlay spectrum sharing* and *underlay spectrum sharing*.

In overlay spectrum sharing, nodes access the network using a portion of the spectrum that has not been used by PUs. SUs in spectrum overlay will only use the licensed spectrum when primary users are not transmitting. Therefore, there is no interference temperature limit imposed on secondary user's transmission. Instead, SUs need to sense the licensed frequency band and detect the spectrum holes, in order to avoid interference to primary users.

On the other hand, in underlay spectrum sharing, the spread spectrum techniques are exploited such that the transmission of a CR node is regarded as noise by PUs. In this case, SUs are allowed

to transmit their data in the licensed spectrum band when primary users are also transmitting. The interference temperature model [7] is imposed on transmission power of SUs, so that the interference at a primary user's receiver is within the interference temperature limit and PUs can deliver their packet to the receiver successfully. Spread spectrum techniques are usually adopted by SUs to fully utilize the wide range of spectrum. However, due to the constraints on transmission power, SUs can only achieve short-range communication. If PUs transmit data all the time in a constant mode, spectrum underlay does not require SUs to perform spectrum detection to find available spectrum band.

2.2.2 Cooperative Sharing vs Non-cooperative Sharing

Spectrum sharing can be *cooperative* or *non-cooperative* ([2], [45]). In cooperative spectrum sharing, the effect of the communication of one node on other nodes is considered. A common technique used in these schemes is forming clusters to share interference information locally. This localized operation provides an effective balance between a fully centralized and a distributed scheme. On the other hand, only a single node is considered in non-cooperative schemes. As interference in other CR nodes is not considered, non-cooperative schemes may result in reduced spectrum utilization. on the other hand, they do not require frequent message exchanges between neighbours as in cooperative solutions.

2.3 Algorithmic Viewpoint

From the algorithmic point of view, the most common techniques [7] of dynamic spectrum management in CRNs can be classified into several categories such as graph theory based approaches, heuristic approaches, evolutionary approaches, game theory based approaches, etc. We now have a brief discussion on these categories.

2.3.1 DSA Approaches Based on Graph Theory

Graph Theory based approaches ([2], [7], [13]-[19]) are most commonly used DSA technique. In such approaches, a network is visualized as a graph where the vertices correspond to CR devices and edges correspond to connections among them.

Among the graph-based techniques used, the most common one is based on constructing *the network conflict graph* [7] that captures the interference between neighbouring SUs ([13], [14], [20]).

We provide an illustration of generating network conflict graph for a particular topology, in Figure 2.3.

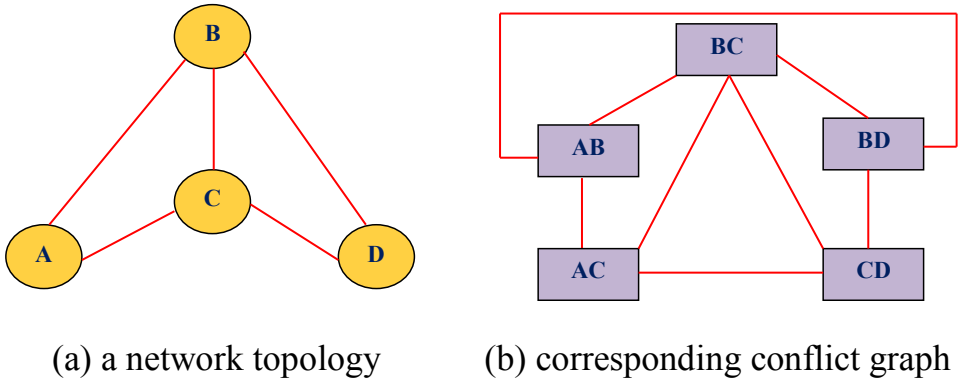
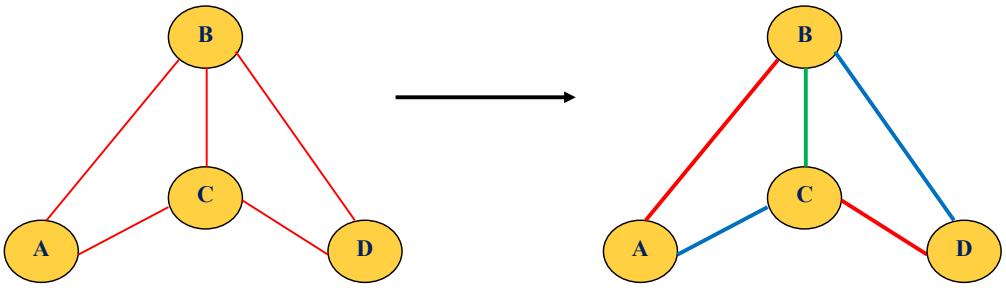


Figure 2.3: Generating network conflict graph (source: [7])

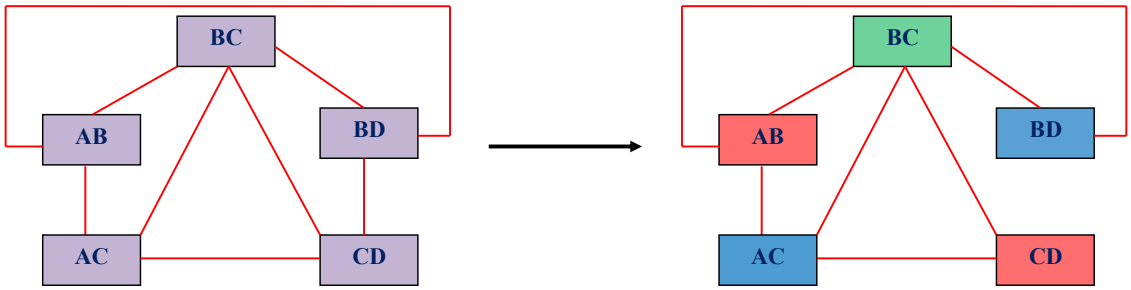
A conflict graph can be simple, weighted, multi-point or dynamic. A first step is to form the connectivity graph, which shows the connectivity and the communication between the network nodes.The vertices of a conflict graph correspond to the links between the nodes and the edges are drawn between links (vertices) that can interfere with each other when assigned the same or adjacent spectrum bands. In weighted conflict graphs, the weights on the edges represent the interference model or the required channel separation between the links. Multi-point conflict graphs can be used to simplify the conflict graph in cases where a single SU is transmitting to multiple receivers. Another approach uses dynamic conflict graphs to capture the possible changes in the interference due to the assignment produced in each step. Dynamic conflict graphs are formed at each step of the DSA algorithm and take into account the aggregated interference effect ([16], [17]).

Conflict graphs are commonly used in centralized approaches where the spectrum server constructs the graph and assigns channels to the links of the graph. In distributed approaches, the SUs themselves form the sets of available channels and negotiate with their neighbors while selecting spectrum bands in order to avoid interference between the links and maximize their performance.

Graph coloring is also used as a DSA algorithm ([15], [18], [21]), where the DSA problem is mapped into a graph coloring problem. The colors can be either at the vertices or at the edges, representing the spectrum bands that are assigned to the SUs or the links respectively. To be specific, we can apply edge-coloring to the network topology graph and vertex coloring to the network topology graph, as demonstrated in Figure 2.4. Here, different colors represent different spectrum bands that are assigned



(a) Edge coloring of the network topology graph



(b) Vertex coloring of the network conflict graph

Figure 2.4: Use of graph-coloring in DSA.

to corresponding edges or vertices.

In addition, few previous works constructed a bipartite graph ([16], [19]) with the available spectrum bands on one side, the SU connections on the other side, as demonstrated in Figure 2.6. Here, the SU connections can be formulated using the network conflict graph. Another approach [46] construct a layered graph to model the cognitive network is proposed. This layered graph models the channel information at each node and shows the interconnection between channel assignment and routing paths, resulting in much easier procedures for shortest path search.

2.3.2 DSA Approaches Based on Game Theory

Another important class of approaches for DSA is based on game-theory ([22]-[27]). A DSA game usually has three sets of elements: the players, the action space, and the utility function(s). The basic model of game theory based DSA is $G = \{n, S_i, U_i\}, i \in n$; where n is the number of players. S_i is the

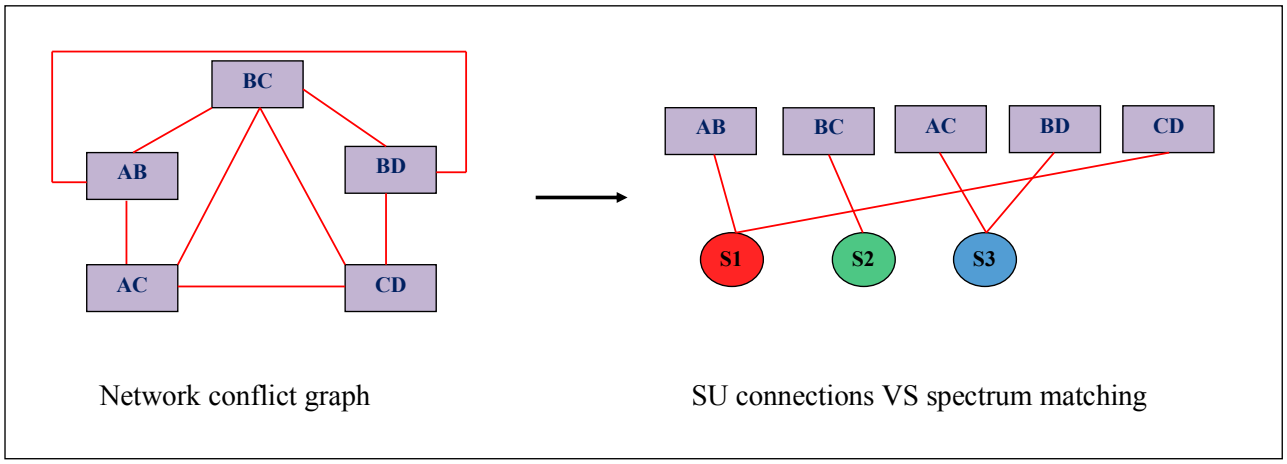


Figure 2.5: Bipartite matching between SU connections and available spectrum fragment for DSA.

strategies of players, which is perceived as the actions. U_i is the set of utility function that the players associate with their strategies. There are two types of games, cooperative and non-cooperative, based on whether the players (i.e., CR users) exchange information regarding their decisions or not.

In game theory based DSA, players are usually the SUs that take part in the game and contend for channel access. PUs can also be active players, although their sets of frequencies may be constant and used only to avoid being selected by the SUs. The players have a set of utility functions (U_i), which is the set of available frequency bands and the action space is the cartesian product of the sets of actions of all players. Moreover, each player has a utility function that is used to translate the action space into the real world needs, namely the frequency bands to meet the SU requirements. The objective is to maximize each SU's utility function, by taking into account the impact of its decisions on the other players. For games with specific characteristics, a steady state performance equilibrium always exists, and any unilateral change of a player results in a lower utility for that player. This solution is called the Nash Equilibrium.

Auction theory [24], which can be considered as a specific branch of game theory, has also been applied spectrum management purpose. Spectrum auctions or spectrum markets have been widely studied in the literature as a solution for the spectrum assignment problem ([22], [25]). The SUs contend for the same channels and a regulator conducts an auction to sell the rights on a set of channels to PUs and SUs.

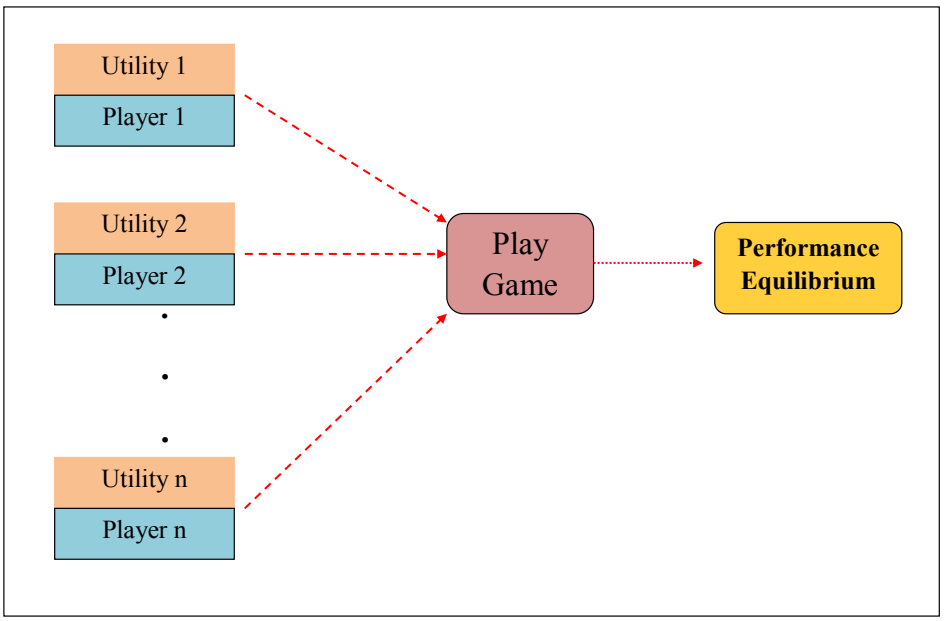


Figure 2.6: Basic components of a cognitive game.

2.3.3 DSA Approaches Based on Heuristics and Evolutionary Algorithms

As DSA problem belongs to the class of NP-complete problems, there is no known algorithm that can generate a guaranteed optimal solution in polynomial time [32]. To address this issue, heuristic and evolutionary algorithms are widely used to speed up the process and find a good (optimal or near-optimal) solution quickly.

Heuristic search algorithms ([18], [21], [32], [33]) are often used in DSA problem. The design of heuristic methods include four major components: initial solution, set of actions, heuristic function, and final solution. Such methods are typically iterative, where we start searching from a *initial solution*. This initial solution is often called the *initial state*, which is generated randomly in most cases. Then, in an iterative manner, we perform *set of actions* to this initial solution to generate better solutions. The optimality of a solution is determined by a heuristic function, which must be carefully designed to perfectly evaluate the goodness of a particular solution. We direct our search in an iterative manner in quest of reaching the globally optimal solution. This globally optimal solution is our *final solution*, which is often termed as the *goal state*.

Evolutionary algorithms (EAs) are also used in DSA ([34]-[37]), as they are proved to be very efficient in finding best solutions in multidimensional optimization problems. Genetic algorithms [35], swarm intelligence [36], and ant colony optimization [37] are the most common EA approaches

that are used in dynamic spectrum management. In a typical EA, the solution of the problem is first represented as *chromosomes*. Chromosomes are collected in groups called *population*. The fitness of these chromosomes are evaluated using a *fitness function*. Then, in each iteration, a set of chromosomes (i.e., parents) are selected for breeding according to a *selection criteria*. Next, *crossover* and *mutation* techniques are applied to those parents to generate new chromosomes (i.e., off-springs). Finally, the population is updated with the newly generated off-springs in an iterative manner. Fitness function judges the goodness of a solution and directs the *search* accordingly. Good solutions satisfy the interference constraints or, in general, the requirements of a good spectrum assignment.

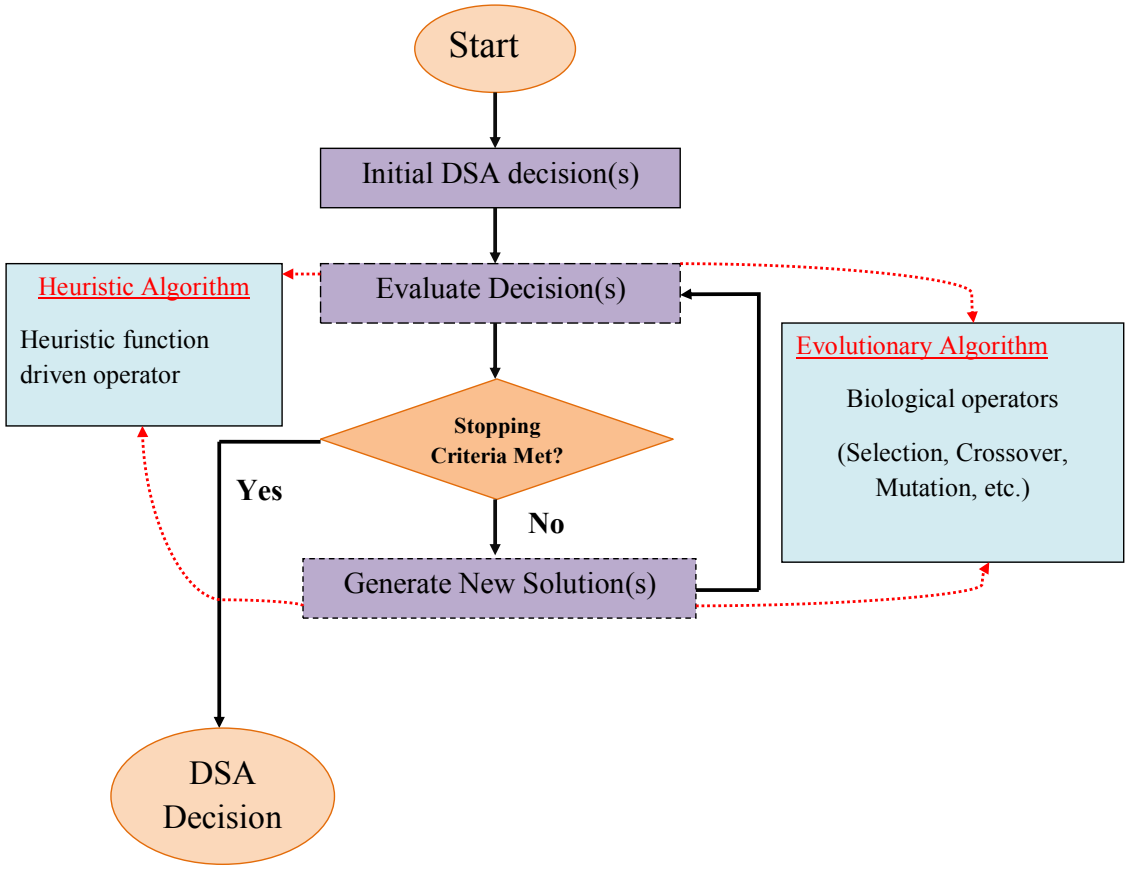


Figure 2.7: Basic structure of heuristic and evolutionary algorithm based DSA

Basic structure of heuristic and evolutionary algorithm based DSA is illustrated in Figure 2.7. The underlined methodology is same as shown in this figure. Main difference is within the operators they use to drive their search. Heuristic methods use heuristic function driven operators, whereas, evolutionary algorithms use biological operators as mentioned above.

The advantage of heuristic and evolutionary approaches is that, they are simple, easily to implement, and they can find a near-optimal solution at a reasonable computational overhead. Furthermore, they tend to be less sensitive to variations in problem characteristics and data quality [47]. A disadvantage of of such approaches is that, there is no analytical methodology to explain their convergence properties and they get stuck in local optimal solutions, which can be far from the global optimal solution.

2.3.4 Other DSA Approaches

In addition to the above mentioned approaches, approaches based on linear programming ([16], [30]) are also common for DSA in CRNs. The joint power/rate control and spectrum allocation problem can be formulated as a Mixed Integer Non-Linear Programming (MINLP) problem [48]. Furthermore, as MINLP problem is NP-hard, it is often transformed into a Binary Linear Program (BLP), containing only binary parameters with linear objective function and constraints. This transformation is possible because wireless communication systems are assumed to have a finite number of available channels (each one with a specific maximum power constraint) and multi-rate capability of the SUs is discrete by nature. The transformation from MINLP into BLP is performed because it has a uni-modular constraint matrix which can be solved in polynomial time using standard linear programming (LP) techniques.

A number of fuzzy logic based DSA ([31], [49]) have been proposed in recent studies as well. A Fuzzy Logic Controller (FLC) consists of four modules: a fuzzy rule base, a fuzzy inference engine, and a fuzzification/defuzzification module. The fuzzy rule base consists of a set of rules, which can be based on prior knowledge, questionnaires, or SU measurements.

In addition, DSA approaches based on Q-learning [31], single agent learning [28], multi-agent learning [29], etc. are recently being used in dynamic spectrum management for CRNs.

2.4 Characteristics of Our Approach

We have discussed the advantages and disadvantages of these approaches and motivation of our approach in the previous chapter. We have used GALS for DSA, which, according to the discussion above, has the following characteristics:

- It is a distributive approach as SU make their own channel assignment decisions.

- It is cooperative as the SUs exchanges information among them.
- It uses underlay spectrum sharing mechanism.

We have chosen a distributed approach because it is more challenging to design an efficient DSA algorithm for distributed CRNs. Besides, fairness improvement in distributed DSA is still a major issue that has been repeatedly addressed in the literature. Besides, future pragmatic deployments of CRNs are mostly proposed for distributed and cooperative cellular networks. We tried to use the most prospective and challenging network architecture, spectrum sharing mode, and other network parameters in our approach. Next, we discuss the details of our network model explaining how we incorporated these characteristics into our simulation environment. Besides, we also mention the CRCN simulator modifications as well. We present all these issues in the next chapter.

Chapter 3

Network Model

We already discussed the basic architecture of CRNs in the previous chapter. We also mentioned how DSA enables the SUs and PUs to share a set of spectrum bands. We have used CRCN simulator, which is based on `ns-2`, to design our CRN environment. In this chapter, at first, we present the network model that we follow in our work. Then, we point out the modifications we performed in the basic CRCN simulator to implement such CRN environments.

3.1 Our Network Model

We consider a single-radio multi-channel CRN where multiple CR users consisting PUs and SUs share a set of orthogonal channels. The PUs and SUs are static and randomly distributed in a two dimensional area (Figure 3.1). They use a common control channel (CCC) for exchanging messages. A CR user cannot transmit over multiple channels at the same time. Besides, they operate in a half-duplex manner, i.e., they cannot receive while transmitting, and vice versa.

Two adjacent CR users can communicate when they both tune to the same channel. Here, SUs can interfere themselves as long as there is enough power for transmission. For data transmission, two operational constraints must be met:

1. The total amount of interference caused by all SU transmissions to each PU must not exceed a predefined threshold (i.e., underlay spectrum sharing).

2. For each CR user, the received signal to interference plus noise ratio (SINR) must exceed a predefined threshold.

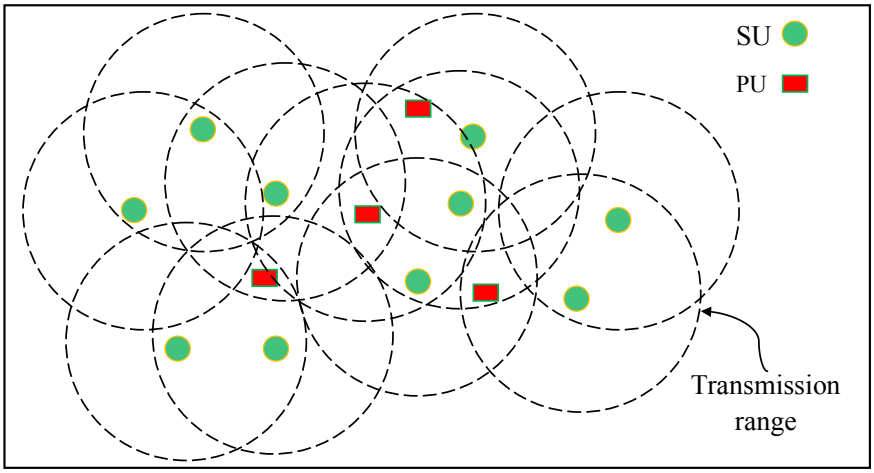


Figure 3.1: Sketch map of our CRN model

CR users make their own channel assignment decisions, and thus the channel assignment decision process is distributive. Each CR user maintains a list of currently active CR users and their corresponding channels. Each active CR user periodically broadcasts a *strategy packet*[1] using the CCC. On receiving strategy packets, CR users periodically update their individual lists. These lists help an SU in keeping track of the current network scenario. When an SU needs to access a channel for data transmission, DSA algorithm finds the best channel according to the current network scenario.

For performance evaluation, we generate varied CRN scenarios using this network model. Then, we simulate these network scenarios to evaluate the performance of different DSA algorithms. In our simulation, we use CRCN simulator [39], which requires few modifications to mimic the real CRN environment. Next, we discuss these modifications which we performed in the basic CRCN simulator.

3.2 Simulator Modifications

We use CRCN simulator [39], the most widely used discrete event simulator for CRNs. It is based on `ns-2`. The basic CRCN simulator provides complete single-radio multi channel functionalities for CRNs. It also provides all the interference information associated with each user, over each channel. However, it does not incorporate the complete presence and interference caused by PU activities. Besides, spectrum sharing and spectrum mobility features are also incomplete in the basic simulator. These features are very important to mimic real CRN operation. In addition to these features, few

[1]Strategy packet contains information about the user and the channel it is currently using for data transmission.

additional modifications are required to support the complete functionality of our network model, which we discussed in Section 3.1. Consequently, we make the following modifications in the basic CRCN simulator.

(i) At first, we utilize an existing data structure *Strategy* (defined in *mobilenode.h*), to track the active users, available channel list, and current transmissions of the network. Each CR user updates its own copy of this data structure by the use of *strategy packet* mentioned in Section 3.1.

(ii) Then, we implement our DSA algorithm and ensure the following constraints:

- When the total amount of interference caused by SU transmissions to a PU on a particular channel reaches a predefined threshold, all SUs within its transmission range remove that channel from their available channel lists, and
- When a PU wants a channel that is currently being used by a SU within its transmission range, the SU stops its current transmission, leaves that channel, and searches for other available channels.

(iii) In addition, we change the maximum limit on the number of channels. The basic simulator sets this limit to 12. We change it to be able to perform simulations with a larger number of channels.

Table 3.1: Modifications made in the basic CRCN simulator

Modifications	**Modified files**	**`ns-2` directories**
(i) and (ii)	mobilenode.h, mobilenode.cc	common
	macng.h, macng.cc	mac
(iii)	phy.h	mac
	topography.h	mobile

We mention the modified files in CRCN simulator in Table 3.1. We have incorporated these features along with other necessary changes to the basic CRCN package, and implemented our own DSA algorithm. For future research purpose in these regard, we are have uploaded the necessary files and stated the necessary operations step by step in [50]. By following these steps, one can download and install the basic `ns-2.31`, CRCN package, and integrate all it's missing features that we discussed above. To the best of our knowledge, few researchers around the world are already using our modified simulator for implementing and testing their own DSA algorithms.

We considered a single radio per channel and static CR users for simplicity. We plan to extend our work for multi-radio and dynamic CRNs in future. Besides, we need further modifications to the CRCN simulator to be able to implement and test dynamic multi-radio CRNs. We will discuss these issues in our future work.

In this chapter, we presented our network model that we follow in our work. We will provide the specifications of the network parameters and simulation settings in Chapter 7. Now, based on this network model, we provide our primary performance evaluation of the basic GA-based DSA in the next chapter.

Chapter 4

GA-based DSA With Basic Genetic Operators

GA is widely used for DSA in CRNs. A basic GA-based DSA procedure is shown in Figure 4.1. First, a solution is represented as a *chromosome*. Chromosomes are collected in groups each called a *population*. The fitness of these chromosomes are evaluated using a *fitness function*. Then, in each iteration, a set of chromosomes (i.e., parents) are selected for breeding according to a *selection criteria*. Next, *crossover* and *mutation* operations are applied to those parents to generate new chromosomes (i.e., off-springs). Finally, the current population is updated with the newly generated off-springs in an iterative manner. In the following subsections, we elaborate all these aspects from the perspective of our approach.

4.1 Chromosome Representation

Suppose, we have a set of n CR users with n_s SUs and n_p PUs (i.e., $n = n_s + n_p$). We assign an unique ID number u_r ($r \in \{1, 2, \ldots, n\}$) to each CR user. These n CR users share a set of m orthogonal channels, $C = \{c_1, c_2, ..., c_m\}$.

We represent a chromosome, which is also known as *an individual*, with an $n-$dimensional vector, denoted as $I =< i_1, i_2, \ldots, i_j, \ldots, i_n >$. Here, $i_j = c$, where $c \in C$, for all active users. If u_j is not currently in transmission (i.e., idle), then $i_j = -1$. Therefore, each individual is created in the following manner: using the maintained lists, the SU finds the active users. Then, for each user u_j

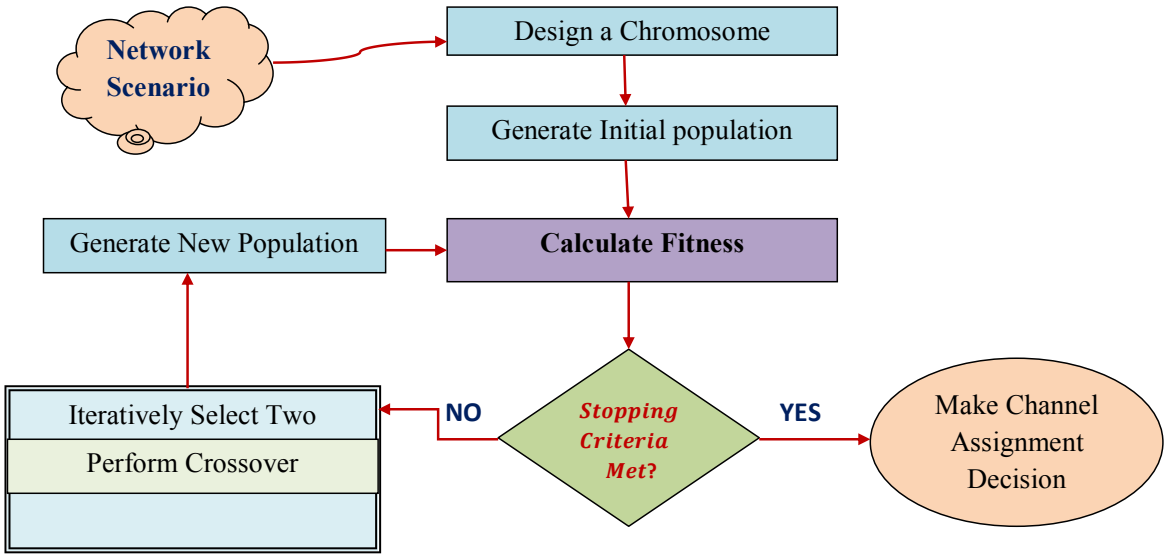


Figure 4.1: Basic GA-based DSA

(including itself), it assigns current channels $c \in C$ or -1 to i_j for active and idle users respectively.

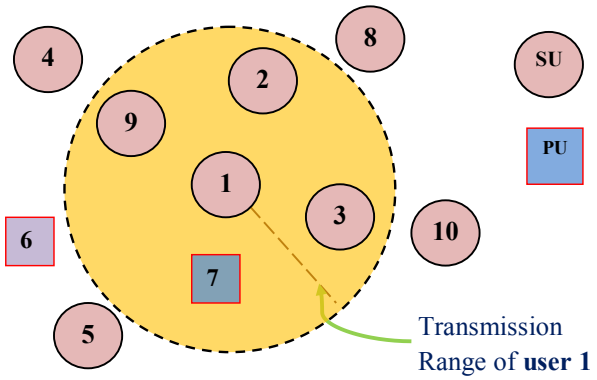
Figures 4.2 illustrates an example of our chromosome formulation. There are 10 CR users in the network shown in Figures 4.2(a). Here, user 5 and 10 are idle, and therefore, their corresponding bit-positions are assigned -1. On other bit-positions, we choose a random number between 1 and $m = 4$, as their corresponding users are active. This is how we create a chromosome in our approach. A set of such chromosomes form our initial population.

4.2 Fitness Function Formulation

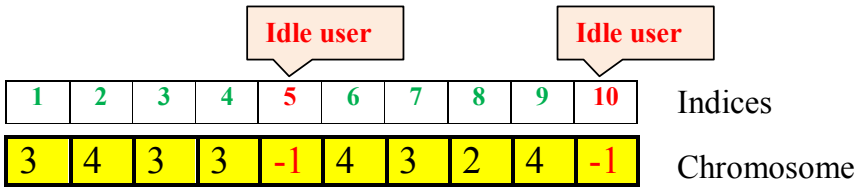
Next, we need to adopt an objective criteria to define the fitness of an individual. There are several objective criteria [7] for DSA in CRNs. We adopt two most challenging ones, interference and fairness, as our criteria. Here, we want to minimize the total number of interferers among SUs over the whole network ensuring that no SU interferes with the PUs[1]. In addition, we want to achieve fairness through leading towards maximum channel utilization.

Let us consider a candidate solution, I, for an SU u_j ($j \in \{1, 2, \ldots, n\}$). Let $IN(I)$ refer to the total number of interferers (both SUs and PUs) of u_j according to solution I. Therefore, $IN(I) =$

[1]The constraint is to keep the total amount of interference to a PU caused by SU transmissions on a particular channel below a predefined threshold.



(a) A sample network scenario (n = 10)



(b) Chromosome formulation for user 1 (m = 4)

Figure 4.2: Chromosome formulation in our approach

$INS(I) + INP(I)$, where $INS(I)$ and $INP(I)$ refer to the number of interfered SUs and PUs, respectively. Using these definitions, we design a penalty function for I as follows:

$$P(I) = [2 + IN(I)]^{[1+INP(I)]} \tag{4.1}$$

Here, $P(I)$ penalizes I for interfering any CR users, however, interfering PUs causes significantly more penalty than interfering SUs. Therefore, lower values of $P(I)$ refer to good solutions.

In addition to interference, we also consider fairness in channel utilization in our fitness function. To do so, we utilize the interference information associated with a CR node over each channel. Let c ($c \in C$) be the channel that solution I suggest for u_j. Then, we denote $ChanIN(c, u_j)$ as the interference associated with u_j over channel c. We normalize this interference value using the following equation:

$$f(c, u_j) = 1 + \frac{ChanIN(c, u_j)}{\sum_{i=1}^{m} ChanIN(c_i, u_j)} \tag{4.2}$$

Finally, we design our fitness function $F(I)$ for candidate solution I, as follows:

Table 4.1: All possible ranges of values of F(I) for $n = 6$ and $n_p = 2$

$INP(I)$	$IN(I)$	**Range of** $F(I)$	Range of $F_1(I)$
0	0	**[1, 2]**	[1, 2]
	1	**[1.58, 3.17]**	[2, 4]
	2	**[2, 4]**	[3, 6]
	3	**[2.32, 4.64]**	[4, 8]
1	1	**[3.17, 6.34]**	[4, 8]
	2	**[4, 8]**	[6, 12]
	3	**[4.64, 9.2]**	[8, 16]
	4	**[5.17, 10.3]**	[10, 20]
2	2	**[6, 12]**	[9, 18]
	3	**[6.97, 13.93]**	[12, 24]
	4	**[7.75, 15.5]**	[15, 30]
	5	**[8.42, 16.84]**	[18, 36]

$$\begin{aligned} F(I) &= f(c, u_j) \times \log_2[P(I)] \\ &= f(c, u_j) \times [1 + INP(I)] \times \log_2[2 + IN(I)] \end{aligned} \tag{4.3}$$

4.2.1 Interpretation of Our Fitness Function

In Equation 4.3, we use logarithms to scale the values of the penalty function $P(I)$. Here, the minimum possible value of $\log_2[P(I)]$ is 1, which is pertinent for $IN(I) = 0$. Besides, by definition, $f(c, u_j) \in [1, 2]$. Therefore, the minimum possible value for $F(I)$ is 1.

Now, to visualize $F(I)$, we consider a simple network scenario with 4 SUs and 2 PUs (i.e., $n = 6$, $n_p = 2$, and $n_s = 4$). We show all possible range of values that $F(I)$ can produce in Table 4.1. As Table 4.1 demonstrates, $F(I)$ produces overlapping ranges of values for different $[INP(I), IN(I)]$ pairs. These overlapping ranges are resolved by the fairness component $f(c, u_j)$. For instance, let us consider two solutions I_1 and I_2, where $[INP(I_1), IN(I_1)] = [0, 1]$ and $[INP(I_2), IN(I_2)] = [0, 2]$; the channel suggested by I_1 and I_2, is c_1 and c_2, respectively. As shown in Table 4.2, values of $F(I_1)$ and $F(I_2)$, span over [1.58, 3.17] and [2, 4], respectively, having an overlap over the range [2, 3.17]. Therefore, although solution I_1 interferes less SUs than solution I_2 (i.e., $P(I_1) < P(I_2)$), it is possible to get $F(I_1) > F(I_2)$, based on the values of $f(c_1, u_j)$ and $f(c_2, u_j)$. Table 4.3 shows the overlapping and non-overlapping ranges in terms of different conditions on $f(c_1, u_j)$ and $f(c_2, u_j)$.

Furthermore, we illustrate the values of $F(I)$ for all possible $[INP(I), IN(I)]$ pairs in Figure

4.3(a). This illustration indicates that, for a particular value of $INP(I)$, as increase in number of SU interferers causes a monotonous increase in $F(I)$. To demonstrate this smooth and consistent increase in $F(I)$ over all possible $[INP(I), IN(I)]$ pairs, we compare $F(I)$ with another potential function $F_1(I) = f(c, u_j)(1 + INP(I))(1 + IN(I))$. Here, $F_1(I)$, or functions similar to this are intuitive choices while designing fitness functions in our case. However, such functions may misrepresent the fitness of a solution. For instance, as Table 4.1 demonstrates, $F_1(I)$ can produce exactly same range of values for two different $[INP(I), IN(I)]$ pairs. For instance, $F_1(I)$ produces range [4, 8] for both $[INP(I_1), IN(I_1)] = [0, 3]$ and $[INP(I_2), IN(I_2)] = [1, 1]$. Besides, as Figure 4.3(b) illustrates, increase in $F_1(I)$ values over different $[INP(I), IN(I)]$ pairs, is not as smooth and consistent as $F(I)$.

Table 4.2: Overlapping range of $F(I_1)$ and $F(I_2)$

Solution	**[INP(I), IN(I)]**	**F(I)**	**Overlapping Range**
I_1	[0, 1]	[1.58, 3.17]	[2, 3.17]
I_2	[0, 2]	[2, 4]	

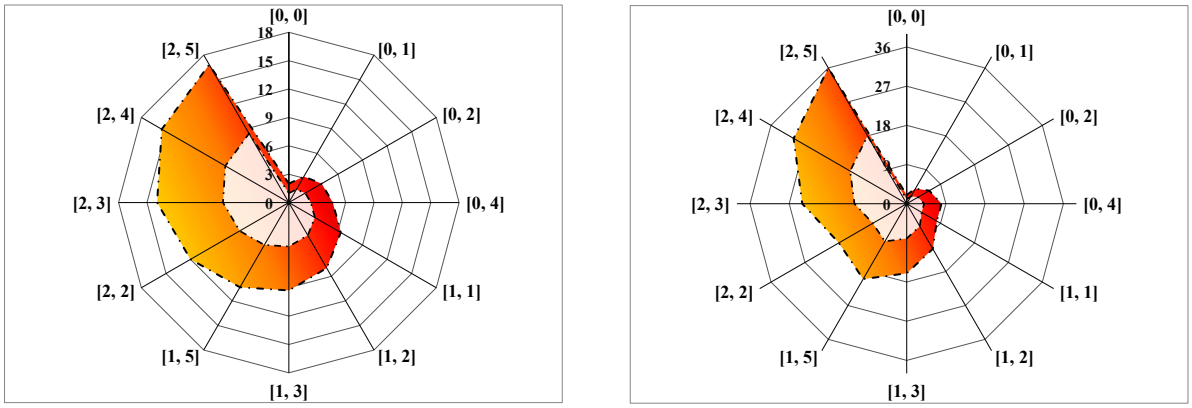
Table 4.3: Effect of different values of $f(c, u)$ on $F(I)$

Condition	**Range**	**Effect on $F(I_1)$ and $F(I_2)$**
$f(c_1, u_j) < 1.26$	Non-overlapping	$F(I_1) < F(I_2)$
$f(c_2, u_j) > 1.585$		
Otherwise	Overlapping	vary based on $f(c_1, u_j)$ and $f(c_2, u_j)$

To summarize the above discussion, $F(I)$ maps interference and fairness information of a candidate solution I to numeric values, where lower values of $F(I)$ (i.e., dark red regions in Figure 4.3(a)) refer to good solutions. Consequently, the channel assignment problem reduces to a function minimization problem where we search for a solution I that minimizes $F(I)$.

4.3 Genetic Operators and Parameter Values

Next, we present genetic operators such as selection strategy, crossover, and mutation operation. In addition, we also choose values of different parameters used in these operators. Examples of the parameters include crossover rate, mutation rate, population size, maximum number of iteration, etc.



(a) Ranges of $F(I)$ (b) Ranges of $F_1(I)$

Figure 4.3: Value ranges of different fitness functions for varying combination of [INP(I), IN(I)] pairs considering $(n = 6, n_p = 2)$

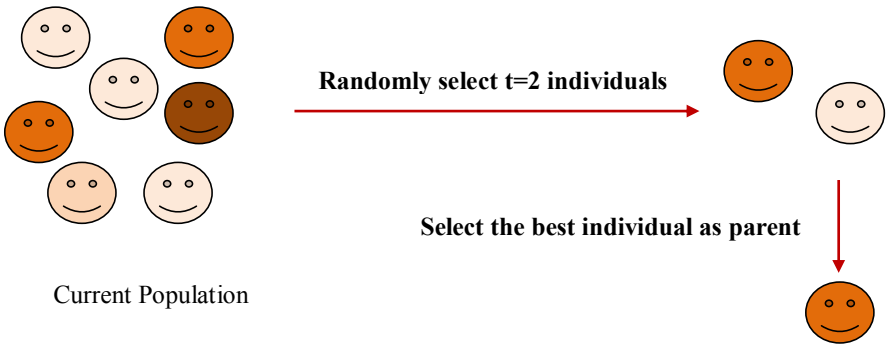


Figure 4.4: Tournament selection mechanism with size $t = 2$

4.3.1 Selection Strategy

For breeding, two individuals are selected in an iterative manner according to a selection criteria [51]. We investigate the performance of two selection strategies: *tournament selection* and *rank-based roulette wheel selection*, which generally exhibit good performance in optimization problems ([51], [52]) similar to DSA.

Let us consider a population φ having $|\varphi|$ individuals. In tournament selection, in each round, we randomly select t individuals from φ. Then, we select the individual with the best fitness among them. We select $t = 2$, i.e., binary tournament, which is the most common in practice. On the other hand, in rank-based roulette wheel selection, we sort all individuals of φ in ascending order of their fitness values. After sorting, the best individual (i.e., the one with the least fitness) goes at position

1 and the worst individual goes at position $|\varphi|$. Then, we find the ranks of all individuals as:

$$Rank(I) = 2 \times \frac{|\varphi| - Position(I)}{|\varphi| - 1}, \forall_{I \in \varphi} \tag{4.4}$$

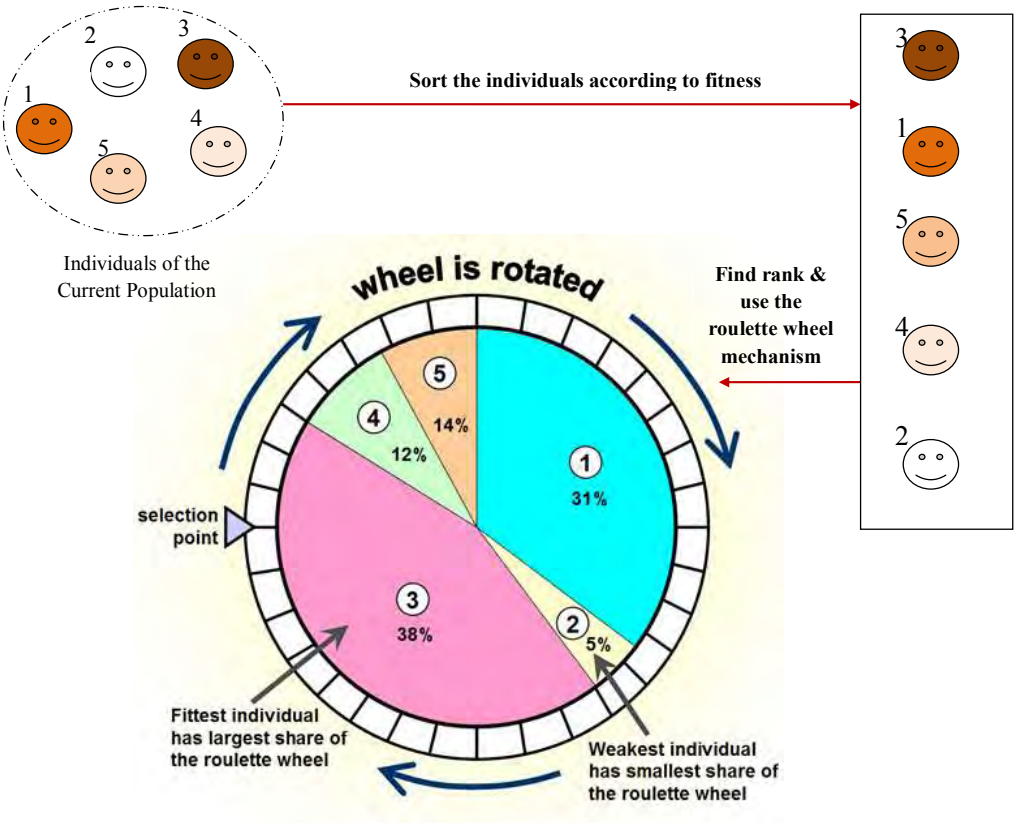


Figure 4.5: Rank-based roulette wheel selection mechanism

$$P_{Selection}(I) = \frac{Rank(I)}{\sum_I Rank(I)}, \forall_{I \in \varphi} \tag{4.5}$$

Here, rank of an individual determines its chance of being selected for mating. We determine this selection probability of an individual using Equation 4.5. Then, for selecting an individual, in each round, we adopt roulette wheel mechanism [51] governed by this equation.

We illustrate tournament selection mechanism in Figure 4.4 and rank-based roulette wheel selection mechanism in Figure 4.5. Here, we consider a simple example with only a few individuals. In addition, we assume the darker colors represent better individuals in the population. As Figure 4.4 shows, in tournament selection, we randomly select $t = 2$ individuals at each round. Then, we select the best individual among them as a parent. On the other hand, Figure 4.5 shows the rank-based roulette wheel

selection mechanism. Here, we numbered the individuals of the population to track their positions in the wheel. At each round, first, we sort the individuals according to their fitness value. Then, we rank them and place them in a roulette wheel. Next to that, we *rotate* the wheel and select a parent which is pointed by the *selection point*.

4.3.2 Tweaking Operators

After the selection of the two parent individuals, we apply crossover and mutation to produce new individuals (also called off-springs). The new off-springs replace their respective parents, and thus subsequently form a new population for the next iteration.

There are a number of crossover and mutation techniques in the literature ([53], [54]). 1-point (Figure 4.6(b)), 2-point (Figure 4.6(c)), and uniform (Figure 4.6(d)) crossover are the three most widely used crossover techniques in the literature.

In 1-point crossover, first, a random position is generated. Then, the parents exchange their values on either side of that position. As demonstrated in Figure 4.6(b), I_1 and I_2 perform 1-point crossover at a randomly chosen position $P = 6$. On the other hand, in 2-point crossover, two random positions are generated. Then, the parents exchange their values that are within those two positions. In Figure 4.6(c), I_1 and I_2 perform 2-point crossover at point $L = 4$ and $U = 8$. The Uniform Crossover uses a fixed mixing ratio between two parents. Unlike 1-point and 2-point crossover, the uniform crossover enables the parent chromosomes to contribute the gene level rather than the segment level. The mixing ratio is usually taken as 0.5, that is, the offsprings have approximately half of the genes from first parent and the other half from second parent. As demonstrated in Figure 4.6(c), values of I_1 and I_2 are exchanged at random positions to perform uniform crossover.

Lastly, point mutation [53] is the generally used mutation technique for optimization problems. In point mutation, first, one position is randomly generated. Then, the value in that position of the parent is altered with another randomly chosen value. In Figure 4.7, point mutation is performed in parent I at position $P = 6$.

We investigate the performance of all these genetic operators in our primary experimental studies that we present next.

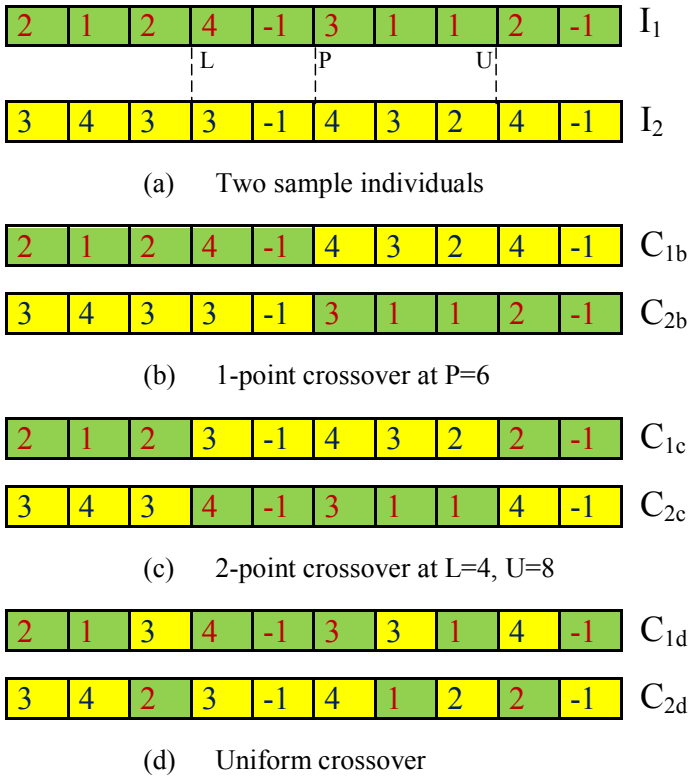


Figure 4.6: Basic crossover operations

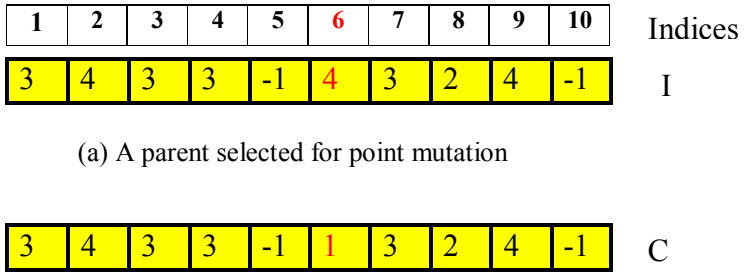
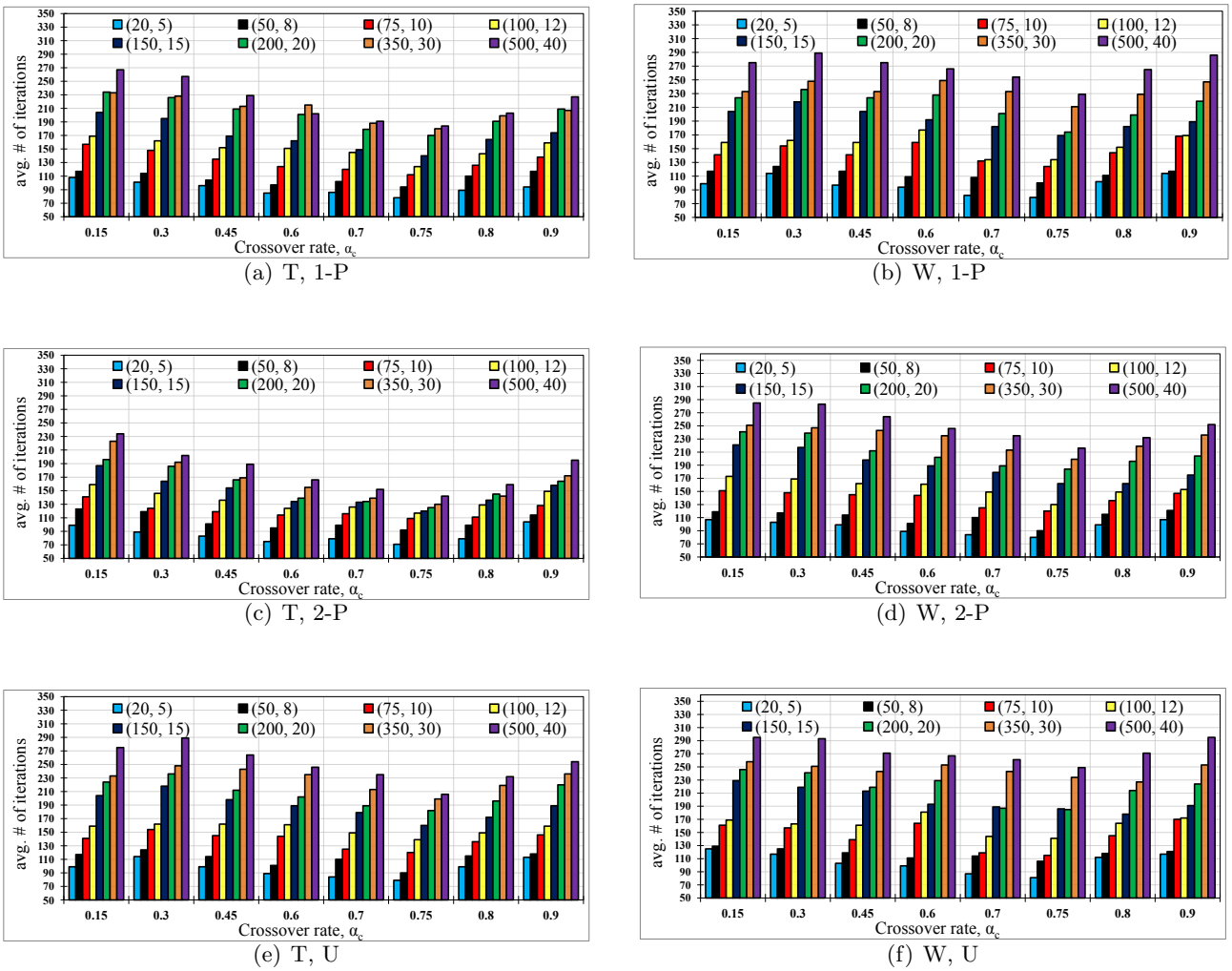


Figure 4.7: Point mutation operation

4.4 Performance of the Basic Genetic Operators

In this section, we present our primary performance evaluation of these basic genetic operators for GA-based DSA in single-radio multi-channel CRNs. We summarize the genetic operators in Table 4.4 and parameter values in Table 4.5. We perform our evaluation over varied network topologies. Here,



(a) T, 1-P (b) W, 1-P

(c) T, 2-P (d) W, 2-P

(e) T, U (f) W, U

Figure 4.8: Performance of GA as DSA algorithm for different (n, m) pairs with different combination of genetic operators and parameter values. Here, 1-P denotes 1-point crossover, 2-P denotes 2-point crossover, U denotes uniform crossover, T denotes Tournament selection, and W denotes Rank-based roulette wheel selection.

we vary the number of CR users, n from 20 to 500 and the number of channels, m from 6 to 40. We present the network topologies with different (n, m) pairs in Table 7.1 (Chapter 7). We present other network parameter values in Chapter 7.

Figure 4.8 demonstrates the performance of GA with different combination of genetic operators and parameter values. Here, we mainly focus on performance in terms of number of iterations needed for convergence. That is, we record the average number of iterations required to reach the global optima solution (i.e., solution I for which $F(I)$ is minimum). If in any case we do not find the global

Table 4.4: Genetic operators for performance evaluation

Operator	**Type(s) Considered**
Selection	Rank-based Roulette Wheel Selection ([51], [52]) (With a linear selection probability function defined in Equation 4.1)
	Tournament Selection ([51], [52]) (With tournament size $t = 2$)
Crossover	1-Point Crossover (Figure 4.6(a)) ([53], [54])
	2-Point Crossover (Figure 4.6(b)) ([53], [54])
	Uniform Crossover (Figure 4.6(c)) ([53], [54])
Mutation	Point Mutation [53]

Table 4.5: Parameter values of the genetic operators

Parameter	**Value(s)**
Population size	75
Maximum number of iterations (M_T)	500
Crossover rate (α_c)	$0.15, 0.30, 0.45, 0.60, 0.75, 0.90$
Mutation rate (α_m)	$0, 1/n, 2/n, 3/n, 5/n$

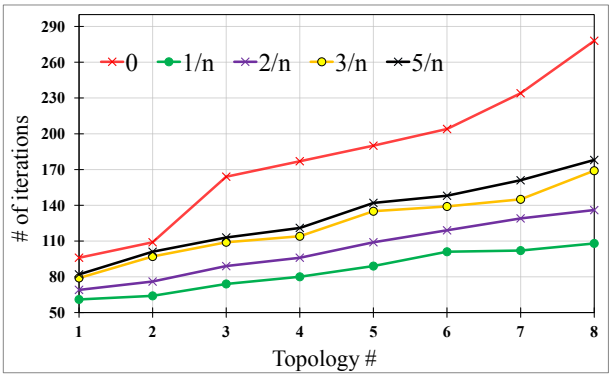
optima solution, we say the algorithm did not converge and discard the case from this part of our evaluation. Therefore, in Figure 4.8, we only consider successful convergences. Each network topology is simulated 20 times for calculating the average values.

Furthermore, in Table 4.6(a), we show the average number of iterations per convergence, averaged over all network topologies, pertinent for all (n, m) pairs in Figure 4.8. Besides, we present the probability of successful convergence, which we get from all simulation runs, in Table 4.6(b). We define *overall convergence rate (C_{rate})* as the ratio between these two convergence parameters, i.e., ratio between the probability of successful convergence and the number of iterations needed for successful convergence. Here, $1/C_{rate}$ gives the expected number of iterations needed to converge to a global optima. Therefore, $1/C_{rate}$ is the rate at which our algorithm leads towards successful convergence in a single iteration. In our evaluation, we calculate (C_{rate}) using the following equation:

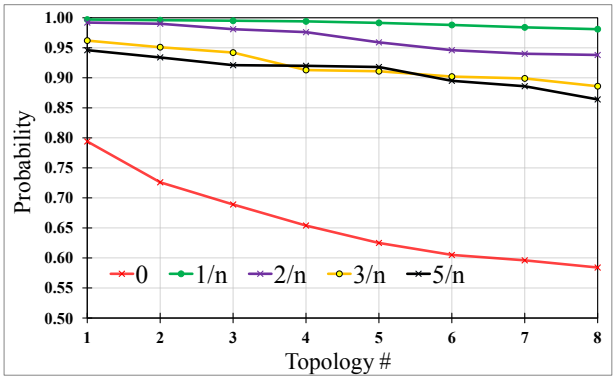
$$C_{rate} = \frac{probability\ of\ successful\ convergence}{number\ of\ iterations\ needed\ for\ successful\ convergence} \times 100 \tag{4.6}$$

Here, we multiply the ratio by 100 to scale up the values. We present the convergence analysis of GA-based DSA with different combinations of basic genetic operators in Table 4.6. As the results presented in Table 4.6 suggest, over all the network topologies considered, GA with 2-Point crossover and tournament selection always outperforms all other combinations. Moreover, with this combination,

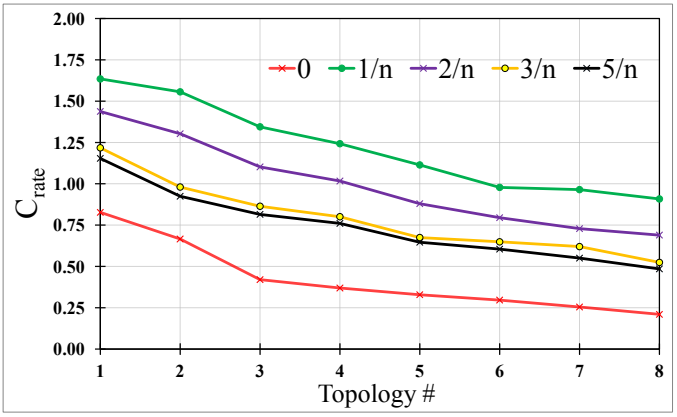
we experience the best performance of GA with a crossover rate, α_c of 0.75.



(a) Avg. number of iterations needed for convergence



(b) Probability of successful convergence



(c) Overall convergence rate (C_{rate})

Figure 4.9: Impact of mutation rate (α_m) on convergence over varied network topologies

Finally, we investigate the impact of changing mutation rate (α_m) in our evaluation. To do so, we choose the best combination of operators found so far, i.e., tournament selection and 2-point crossover with $\alpha_c = 0.75$. With this combination of operators, we measure the performance of GA over varied α_m. The function of α_m is to avoid local optima solutions by introducing diversity in a population as cross-over is not a global operation ([53], [54]). Without mutation (i.e., $\alpha_m = 0$), crossover alone cannot reach global optima in significant number of cases. Besides, our initial choice of $\alpha_m = 1/n$ provides the best performance in terms of convergence.

So far, we found that tournament selection, 2-point crossover with rate 0.75, and point-mutation rate $1/n$ is the best combination of genetic operator for GA in CRNs. Next, we investigate new genetic operators with the optimized parameter values for devising our novel hybrid DSA algorithm.

Table 4.6: Convergence analysis of GA-based DSA with different combinations of basic genetic operators

(a) Average number of iterations needed for convergence

Operators	α_c							
	0.15	0.3	0.45	0.6	0.7	**0.75**	0.8	0.9
T, 1-P	186	178	163	154	145	**135**	153	165
W, 1-P	181	193	181	184	165	**152**	173	188
T, 2-P	**170**	**152**	**139**	**125**	**122**	**113**	**125**	**148**
W, 2-P	193	190	179	170	160	**147**	163	174
T, U	181	193	179	170	160	**146**	164	179
W, U	201	195	183	187	168	**162**	178	192

(b) Probability of successful convergence

Operators	α_c							
	0.15	0.3	0.45	0.6	0.7	**0.75**	0.8	0.9
T, 1-P	0.94	0.94	0.95	0.96	0.96	**0.97**	0.96	0.95
W, 1-P	0.94	0.94	0.95	0.95	0.96	**0.97**	0.95	0.94
T, 2-P	**0.96**	**0.97**	**0.97**	**0.97**	**0.96**	**0.97**	**0.97**	**0.96**
W, 2-P	0.92	0.94	0.95	0.95	0.95	**0.96**	0.95	0.94
T, U	0.94	0.94	0.95	0.96	0.96	**0.97**	0.96	0.95
W, U	0.93	0.94	0.93	0.94	0.95	**0.97**	0.96	0.95

(c) Overall convergence rate (C_{rate})

Operators	α_c							
	0.15	0.3	0.45	0.6	0.7	**0.75**	0.8	0.9
T, 1-P	0.5	0.52	0.58	0.62	0.66	**0.71**	0.63	0.56
W, 1-P	0.51	0.49	0.52	0.52	0.52	**0.63**	0.55	0.5
T, 2-P	**0.56**	**0.64**	**0.78**	**0.78**	**0.79**	**0.86**	**0.78**	**0.64**
W, 2-P	0.47	0.49	0.53	0.57	0.56	**0.63**	0.58	0.54
T, U	0.52	0.48	0.53	0.56	0.6	**0.66**	0.58	0.53
W, U	0.46	0.48	0.5	0.5	0.57	**0.6**	0.54	0.49

Chapter 5

Devising New Genetic Operators

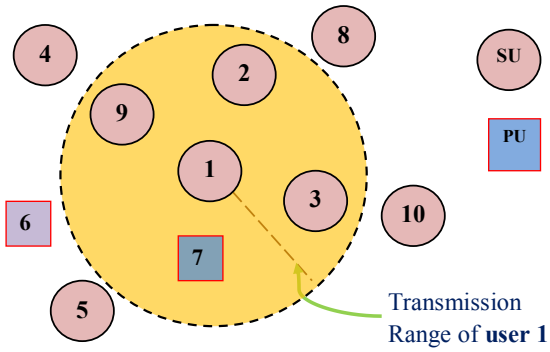
Although GA is highly efficient and scalable for multidimensional optimization problems, we need to select suitable genetic operators and tune parameter values to ensure its best performance for a particular problem ([52]-[54]). DSA, being a multidimensional optimization problem, also demands such selection and tuning. We addressed these issues of selecting best traditional operators and tuning parameter values in the previous chapter. In this chapter, we design two novel genetic operator to improve the performance of a traditional GA-based DSA algorithm. Here, we focus on crossover operation and generation of new population.

5.1 Neighborhood Based Crossover Operation

In our design, fitness of a particular channel assignment mostly depends on channel assignments of its own and that of its neighbors. Here, a good channel assignment causes less interference to its *neighboring* SUs, and no interference to *neighboring* PUs. Therefore, channel assignment of *neighboring* CR users carry most of the information about goodness of a particular assignment. Keeping this in mind, while performing crossover, we exchange the portion of the chromosome that belong only to neighbors of that particular SU rather than exchanging random parts. We name such crossover as neighborhood-based crossover.

Few optimization problems [54] have explored utilizing neighboring information in genetic operators. However, such intuitive approach is yet to be performed and investigated in case of DSA algorithms. In our design, exchanging channel information of only the neighbors facilitate exchanging of more useful information of two individuals. Figure 5.1 illustrates an example of neighborhood-based

crossover for user 1, whose neighbors are user 2, 3, 7, and 9 (indices of CR users are shown in the figure). Here, channel assignments of only these neighboring users in parents I_1 and I_2 are swapped to perform crossover, which produces two off-springs, C_1 and C_2.



(a) A sample network scenario (n = 10)

2	1	2	4	-1	3	1	1	2	-1	I_1
1	2	3	4	5	6	7	8	9	10	indices
3	4	3	3	-1	4	3	2	4	-1	I_2

(b) Two sample individuals (n = 10, m = 4)

2	4	3	4	-1	3	3	1	4	-1	C_1
3	1	2	3	-1	4	1	2	2	-1	C_2

(c) Neighborhood-based crossover for user 1

Figure 5.1: Proposed neighborhood-based crossover

5.2 Local Search Based Survival Selection

In a typical GA, the new off-springs replace their parents and the new population becomes the current population for the next iteration. However, in our approach, we select two *survivors* from the four (two parents and two off-springs) individuals. We illustrate this survival selection process in Figures 5.2. Here, we find two survivors, S_1 and S_2, from two parents (I_1 and I_2) and two off-springs (C_1 and C_2), in the following manner:

- First, we select one individual randomly as our first survivor. This is due to maintain a balance between stochastic and local search. Selecting only the best individuals might limit the diversity

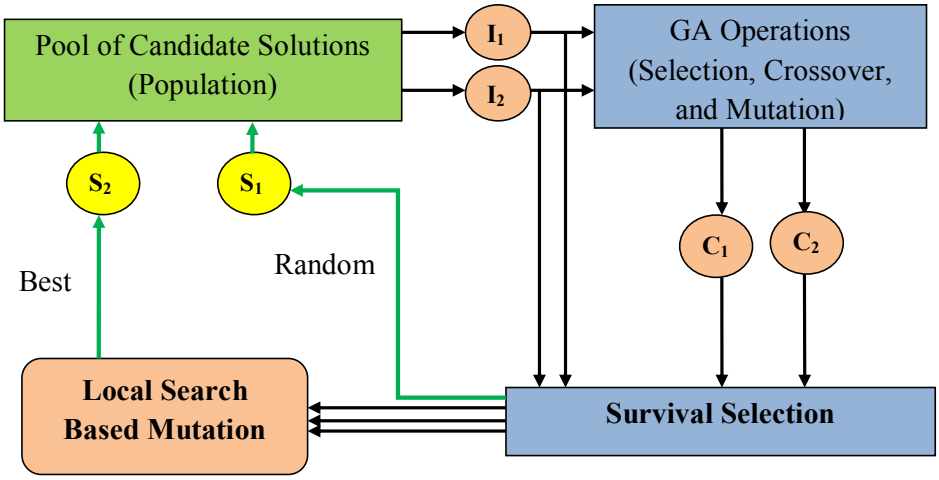


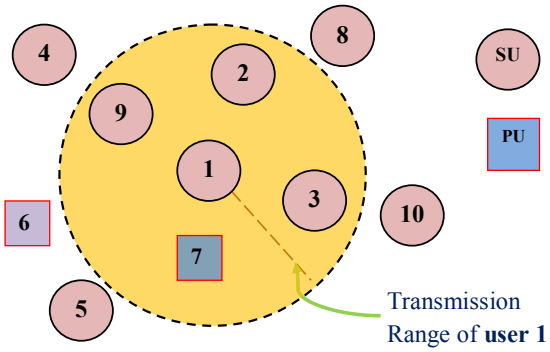
Figure 5.2: Survival selection in our proposed approach

of the population. Keeping this in mind, we balance the stochastic nature of our algorithm by selecting one survivor randomly.

- Then, to select the other survivor, we separately apply local search based mutation on the three candidate individuals left. Here, at first, we find the channel that is suggested for the (local search performing) SU by a candidate individual. Then, we alter this value with c ($c \in C$) such that it minimizes the fitness value of the individual. Finally, we calculate the fitness of the three mutated individuals and select the best one as our second survivor. This time we push the best individual into the next generation.

Figure 5.1 illustrates an example of local search based survivor selection. Here, we consider the same network scenario ($n = 10$ and $m = 4$), that we used for demonstrating neighborhood-based crossover. We consider a candidate parent solution I for user 1. As Figure 5.1 shows, the channel suggested by solution I for user 1, is 3. Now, we perform mutation at position 1 of parent I, to generate $m = 4$ off-springs: C_1, C_2, C_3, and C_4. Then, as we discussed above, we select the best off-spring (having least fitness value) as the survivor. Here, we assume C_2 is the best off-spring, and therefore, select it as the survivor.

Although GA has its own balance between exploitation and exploration, the added deterministic exploration by local search ensures a faster avoidance of possible local optima solutions. This is because, in typical GA, point mutation is the only way of escaping local optima regions. Besides,



(a) A sample network scenario (n = 10)

1	2	3	4	5	6	7	8	9	10	Indices
3	4	3	3	-1	4	3	2	4	-1	I

Channel suggested for user 1

(b) Candidate parent solution for user 1 (m = 4)

1	4	3	3	-1	4	3	2	4	-1	C_1
2	4	3	3	-1	4	3	2	4	-1	C_2
3	4	3	3	-1	4	3	2	4	-1	C_3
4	4	3	3	-1	4	3	2	4	-1	C_4

(c) Mutation on position 1 to generate m = 4 off-springs

2	4	3	3	-1	4	3	2	4	-1	Survivor

(d) Selection of the best off-spring as survivor (here, we assume C_2 has the least fitness value among the four off-springs)

Figure 5.3: Local search based survivor selection

as point mutation is performed on a random position of the individual, it often takes GA a significant number of generations to eventually escape the locally-optimal regions. Therefore, the added deterministic local search based mutation ensures better and faster avoidance of such local optima regions. Additionally, it facilitates converging faster through speeding up the exploitation of GA in globally-optimal regions.

5.3 Performance Evaluation of New Genetic Operators

Now, to determine the effectiveness of our proposed genetic operators, we compare the convergence properties of GALS and GA with the best traditional genetic operators that we found in the previous chapter (Figure 4.8(c) in Chapter 4). We perform this performance comparison in terms of the same convergence parameters. That is, we use average number of iterations needed for convergence, probability of successful convergence, and overall convergence rate. We present the comparison in Table 5.1.

Table 5.1: Performance comparison of GALS and GA with the best combination of operators in terms of convergence

Performance Metric	**GA (T, 2-P)**	**GALS (T, NB)**
Avg. # of iteration needed for convergence	113	64
Probability of successful convergence (%)	0.97	0.99
Overall convergence rate (C_{rate})	0.86	1.55

As Table 5.1 demonstrates, GALS achieves significant performance improvement in terms of convergence. This is exactly what we were looking for. Slow convergence rate and local optima problem are the only concerns of stochastic approaches. Therefore, our hybrid approach, GALS has significantly improved the convergence properties of stochastic DSA. It is to be noted that here we compared the performance of GALS with the best combination of genetic operators that we found in the previous chapter. Therefore, GALS should perform much better than other state-of-the-art DSA approaches. We will provide these performance comparisons in Chapter 7.

So far, we measured performance only in terms of convergence. We will focus on other performance metrics, specially the QoS parameters of the network, later in this paper. Now, having designed such novel efficient genetic operators and tuned parameter values, next, we present the complete algorithm of our proposed approach.

Chapter 6

DSA Procedure Using GALS

In this chapter, we present our proposed channel assignment algorithm (i.e., GALS) and compute its computational complexity. At-first, we present the algorithmic structure of GALS in Figure 6.1. Then, we provide the detailed channel assignment procedure of GALS in Algorithm 1. Finally, we deduce the time complexity and memory requirement of GALS.

As demonstrated by Figure 6.1, the algorithmic structure of GALS is almost similar to a typical GA-based DSA that we discussed in Chapter 4. However, in GALS, we use the novel genetic operators, i.e., neighborhood-based crossover and local search based survivor selection. In Table 6.1, we present the genetic operators and parameter values that we adopt in GALS. We have already discussed the selection strategy and working principal of these operators along with the tuning of parameter values in the previous chapters. Now, we present the complete channel assignment procedure of GALS in Algorithm 1.

Table 6.1: Genetic operators and parameter values adopted in GALS

Operator	**Variant Chosen**
Selection	Tournament Selection ($t = 2$)
Crossover	Neighborhood-based Crossover
Mutation	Point Mutation
Parameter	**Value**
Population size ($\|\varphi\|$)	75
Maximum number of iteration (M_T)	300
Crossover rate (α_c)	0.75
Mutation rate (α_m)	$1/n$

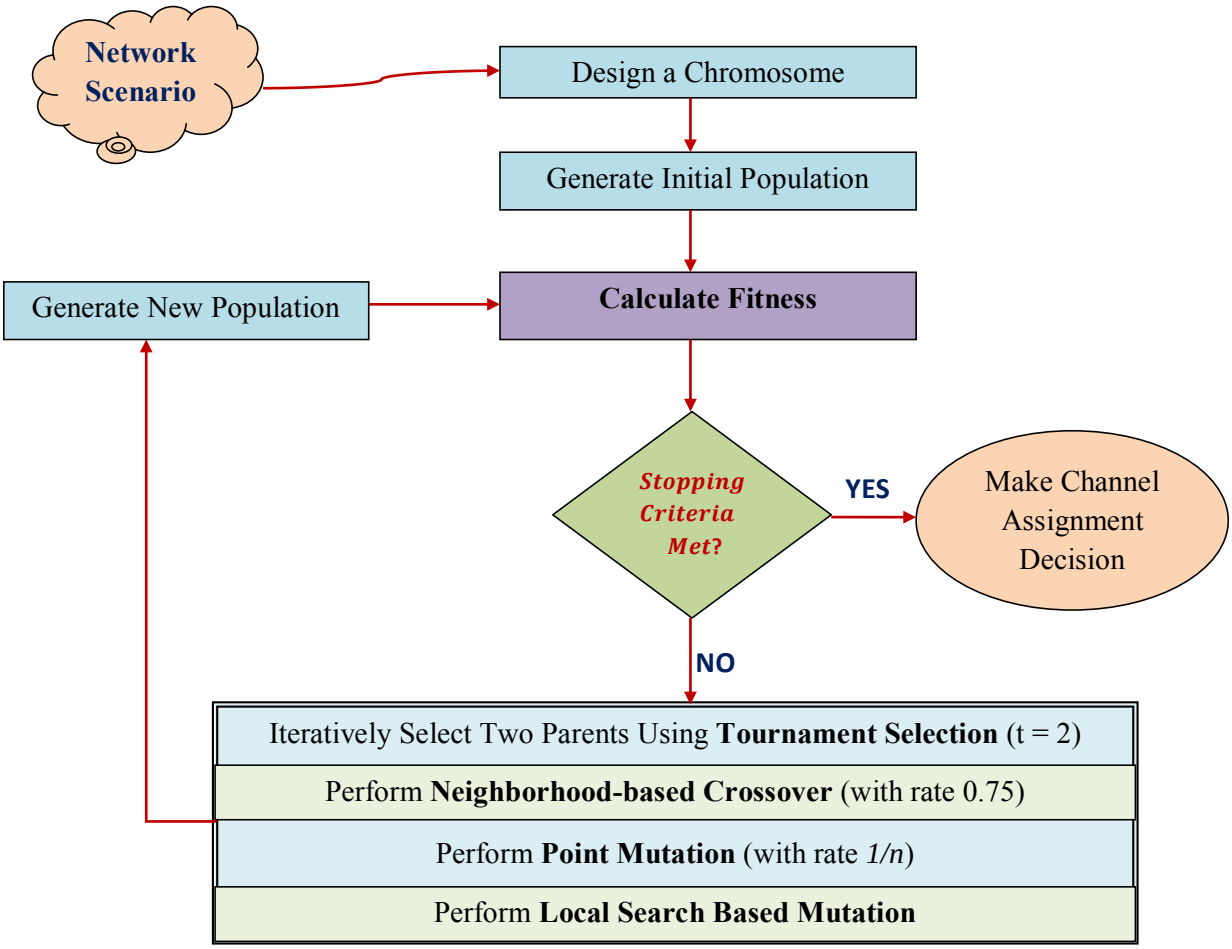


Figure 6.1: Algorithmic structure of GALS

As demonstrated by Algorithm 1, the channel assignment process of GALS starts by checking if there are any unused channels in the network (line 2). If any, we simply choose an unused channel for assignment (line 3). If there are more than one unused channels, then we select one randomly. It is to be noted that, we call a channel *unused*, only if it is not currently being used by any CR users in the entire network. This information is gathered by the use of *strategy packets* that CR users broadcast periodically. We discussed the use and operation of *strategy packets* in our network model (Chapter 3).

If no unused channels are available, then we must select a channel that is most suited for the GALS-calling SU, according to its current network scenario. Here, the SU has only the knowledge about its neighboring CR users and their corresponding transmissions. Besides, it has the channel interference

Algorithm 1: Channel assignment in GALS

```
1  begin
2  if there is an unused channel cl then
3      Select cl and return
4  Initialize parameters
5  Generate initial population φ
6  generation ← 1
7  while generation ≤ M_T & global optima not reached do
8      Initialize new population φ_new = null
9      Calculate fitness of each individual of φ
10     Set S ← null
11     while |φ_new| < |φ| do
12         Select two parents from φ using tournament selection (t = 2)
13         Add the parents to S
14         Perform neighborhood-based crossover (rate α_c)
15         Perform point mutation (rate α_m)
16         Add the generated off-springs to S
17         a ← a random individual in S
18         Remove a from S and add to φ_new
19         foreach individual in S do
20             Perform local search based mutation
21             Calculate fitness
22         b ← the best individual in S
23         Add b to φ_new
24     generation ← generation + 1
25     φ ← φ_new
26 Sort the individuals according to their fitness values
27 solution ← the best individual of φ
28 u_r ← own user ID
29 c ← value at u_r^th position of the solution
30 Select channel c for u_r
31 return
32 end
```

information of each channel (globally available to all CR users). Using this information, the task of GALS is to find the best channel assignment decision by its iterative searching methodology.

We start the iterative searching of GALS by initializing the parameter values for GALS (line 4). We generate initial population for GALS (line 5) as discussed in Chapter 4. This initial population is our first generation of individuals. Then, we perform genetic operations to produce newer generations

Algorithm 2: Local search based mutation

```
1  Input:an individual I
2  begin
3  u_r ← own user ID.
4  ch ← value at u_r^th position of I.
5  for each channel c ∈ C do
6      f_1 ← calculate fitness of I.
7      replace ch with c at u_r^th position of I.
8      f_2 ← calculate fitness of I.
9      if f_1 ≤ f_2 then
10         replace the value c with ch back to u_r^th position of I.
11 return
12 end
```

in an iterative manner (line 7-25).

In each generation, at first, we calculate the fitness of each individual of the current generation (line 9) using Equation 4.3 (Chapter 4). Then, in an iterative manner (line 11-23), we select two parents using tournament selection (line 12). Next to that, we perform neighborhood-based crossover (line 14) with a rate α_c and point mutation operation (line 15) with a rate α_m. These genetic operations produce two offspring individuals from two parents. Then, we select two survivor individuals (line 16-23) from the four individuals (two parents and two offspring), for the next generation. We have already discussed this process in the previous chapter (Figure 5.2). Here, we select one individual randomly as our first survivor (line 17-18). To select the other survivor, we separately apply local search based mutation (line 19-23) on the three candidate individuals left. We present the local search based mutation procedure in Algorithm 2.

As demonstrated by Algorithm 2, for local search based mutation, at first, we find the channel that is suggested for the (local search performing) SU by the candidate individual (line 4). Then, we alter this value with c ($c \in C$) such that it minimizes the fitness value of the individual (line 5-10).

Finally, when GALS receives all three mutated individuals, it selects the best one as the second survivor (line 22-23). This is how the survivors of the current generations iteratively forms the next generation population (line 25).

We perform the genetic operations mentioned above in each generation to direct our search towards global optima region. We terminate our search when we find the globally optimal solution.

Otherwise, we keep searching till the last generation. After terminating our iterative search, we sort[1] the individuals according to their fitness values (line 26). Then we choose the best solution for channel assignment (line 27-30). It is worth mentioning that, according to our survivor selection mechanism, the best individual in any particular generation always survives to the next generation. Therefore, we *remember* the best solution found so far across generations. Consequently, the best solution of the last generation is the best solution of the entire search.

Table 6.2: Time complexity of major computational components of GALS

Particulars	**Time Complexity**
Checking unused channels (line 2)	$O(m)$
Generating initial population (line 5)	$O(n\|\varphi\|)$
Calculating fitness of each individual (line 9)	$O(\|\varphi\|(n+m))$
Tournament selection (line 12)	$O(t)$
Neighborhood based crossover (line 14)	$O(n)$
Point mutation (line 15)	$O(1)$
Local search based mutation (line 16-23)	$O(m)$
Sorting the individuals according to their fitness values (line 26)	$O(\|\varphi\| \log(\|\varphi\|))$
Final channel assignment (line 27-30)	$O(1)$

6.1 Computational Complexity

Evolutionary algorithms are mostly used in optimization problems, which are computationally expensive and suffer from scalability issues when approached by classical methods. Likewise, GA based approaches are also highly scalable and efficient in finding good solutions with reasonable computational overhead. Now, we investigate the computational complexity of our algorithm.

6.1.1 Per-iteration Time Complexity

First, we deduce the *per-iteration* time complexity of GALS based on its algorithmic structure in Algorithm 1. For checking unused channels, we need $O(m)$ time as we have to loop through each channel to see if it is unused. Then, we need $O(n|\varphi|)$ time to generate our initial population consisting $|\varphi|$ individuals, each having n bits. Finding fitness values of each individual costs $O(|\varphi|(n+m))$

[1]We use merge sort [16] in GALS considering its good average and worst case time complexities.

according to our fitness function (Equation 4.3, Chapter 4). Neighborhood based crossover costs $O(n)$ time as we can have at-most $(n-1)$ neighbors in worst case. Local search based mutation costs $O(m)$ time, which is very straight forward from Algorithm 2.

For complexities of other state-of-the art operations, we refer to Table 6.2. In this table, we present the time complexity of major computational components of GALS. Disregarding other constant-time minor operations for simplicity, we deduce the overall time complexity ($T(n,m)$) as follows:

$$\begin{aligned} T(n,m) &\Rightarrow O(m+n|\varphi|) + O(M_T(|\varphi|(n+m)+t+n+m)) + O(|\varphi|\log(|\varphi|)+1) \\ &\Rightarrow O(m) + O(n+m) \\ &\Rightarrow O(n+m) \end{aligned}$$

Here, we deduced the time complexity of GALS by considering the number of iterations constant. This is because, we terminate our search after maximum number of iterations is reached even if we do not reach the global-optima solution. According to our discussion of the convergence properties of GALS in the previous chapter, GALS finds the global-optima solution in 99% cases. Therefore, the number of iteration is not *ideally* constant for GALS (this is true for all evolutionary algorithms). Hence, we call it a *per-iteration* time complexity.

6.1.2 Space Complexity

Next, we calculate the memory requirement of GALS. For storing the status of each channel (i.e., idle or in transmission), and interference information associated with each CR user, we require $O(m+nm)$ memory. Besides, for storing current and next generation individuals (i.e., φ and φ_{new} in Algorithm 1), we need $O(2n|\varphi|)$ space. In addition, for performing crossover and local search based mutation, we require $O(n)$ space. Ignoring few other constant and temporary memory units, we calculate its memory requirement of GALS $S(n,m)$ as follows:

$$\begin{aligned} S(n,m) &\Rightarrow O(m+nm) + O(2n|\varphi|) + O(n) \\ &\Rightarrow O(nm) + O(n) \\ &\Rightarrow O(nm) \end{aligned}$$

According to the discussion above, we found the time complexity ($T(n, m)$) of GALS to be $O(n + m)$, and space complexity ($S(n, m)$) to be $O(nm)$. Here, of-course, n is the total number of CR users in the network and m is the number of channels they share. $T(n, m)$ is linear, which makes GALS a highly scalable DSA algorithm in CRNs. On the other hand, generally we have $m << n$, which leads to $mn << n^2$. Therefore, $S(n, m)$ is more on the linear side rather than quadratic. Consequently, the memory requirement of GALS is also quite reasonable even in cases of large CRNs.

In this chapter, we presented our DSA algorithm GALS in details. Then, we also deduced its computational complexity and we found GALS to be a highly scalable and memory efficient DSA algorithm, which are the basic requirements of distributed CRNs. Now, we focus on investigating the CRN performance using GALS as the underlined DSA algorithm, in terms of several QoS parameters. We provide the detailed performance analysis of GALS in the next chapter.

Chapter 7

Performance Evaluation

In this chapter, we evaluate the performance of GALS algorithm using CRCN simulator. At first, we present the simulation settings under which we perform our simulation. Then, we evaluate the network performance in terms of several QoS parameters. In this primary evaluation, we use network throughput, end-to-end delay, packet delivery ratio, and packet drop ratio as QoS parameters. In terms of these QoS parameters, we present the CRN performance with GALS as the underlined DSA algorithm.

In the second part of our simulation, we compare the performance of GALS with that of several other approach found in the literature. For this performance comparison, we consider several simulation scenarios. Besides, we consider two fairness parameters: channel utilization fairness and per-node fairness, in addition to the above mentioned QoS parameters. Finally, we analyse the simulation results and discuss our findings.

7.1 Simulation Settings

We evaluate and compare the performances of GALS over varying network topologies. We vary the number of CR users (n), number of channels (m), and number of connections per CR user, and data rates in the network to generate different network topologies.

First, we consider different combinations of (n, m) pairs to generate different network topologies, which we present in Table 7.1. For each of these topologies, we evaluate the network performance over varied number of connections per CR user. Then, we compare the performance of GALS with that of several other approach found in the literature. We present the state-of-the-art algorithms that

we consider for performance evaluation in the next section.

For performance comparison of GALS with other state-of the-art algorithms, we generate varied network scenarios in the following manner:

- First, we present the performance evaluation considering 5 CR users per channel. Here, we take the number of connections per CR user to be 1, $n : m$ to be $5 : 1$, and generate different network topologies by varying m from 5 to 30. We present these topologies in Table 7.2.

- Then, we present the performance evaluation with varied n. Here, we fix m to be 10, number of connections per CR user to be 1, and vary n from 20 to 200 to generate different network topologies.

- Subsequently, we present the performance evaluation with varied m. Here, we fix n to be 100, number of connections per CR user to be 1, and vary m from 4 to 32 to generate different network topologies.

- Next to that, we consider the same network topologies (Table 7.1) that we used to evaluate the network performance of GALS. Here, at first, we fix the number of connections per CR user to be 1 and present the performance comparison over all network topologies of Table 7.1. Then, we vary the number of connections per CR user in the network. In that case, we consider a network topology with 100 CR users and 12 channels (i.e., topology 4 in Table 7.1), and vary the average number of connections per CR users from 1 to 6.

- Finally, we investigate the effect of changing the data rate in the network. In all the simulation cases mentioned above, we consider a data rate of 200 packets per second. Here, we vary packets per second from 50 to 500 to investigate the performance of the DSA algorithms. We do so by considering a network topology with 100 CR users and 12 channels (i.e., topology 4 in Table 7.1) and fix the number of connections per CR user to be 1.

Table 7.1: Topology number and corresponding (n, m) pairs

Topology #	1	2	3	4	5	6	7	8
m	5	8	10	12	15	20	30	40
n	20	50	75	100	150	200	350	500

Table 7.2: Different network topologies with 5 CR users per channel

Particulars	Values							
m	5	8	10	12	15	20	25	30
n	25	40	50	60	75	100	125	150

In each cases listed above, we consider the number of PUs per channel to be 2. For example, for $m = 10$, the number of PUs (n_p) is equal to $2 \times 5 = 10$. Throughout our simulation, we run each network topology 30 times and take the averaged value.

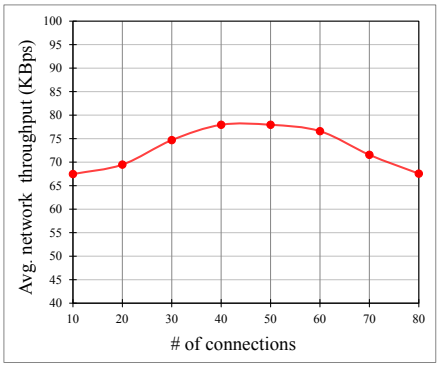
For simulation, we consider a square network coverage area of size 1000m $\times$ 1000m. CR users are randomly deployed across this area following an uniform distribution. The transmission range of each CR user is set to 250m. They use a common control channel (CCC) for exchanging control messages. That is, out of m channels, one (specifically, channel 0 in our simulation) is used as CCC for control messaging, other $(m - 1)$ channels are used for data transmission.

In Chapter 3, we have already presented two operational constraints in our network model. Now, we adopt the parameters pertinent to the constraints. We set the maximum tolerable interference for each PU to 90dBm. Besides, we set the predefined threshold of SINR to 15dB. Additionally, we set other network parameters, such as the path loss exponent to 4 and the noise power spectrum density (N_0) at each SU to -100dBm.

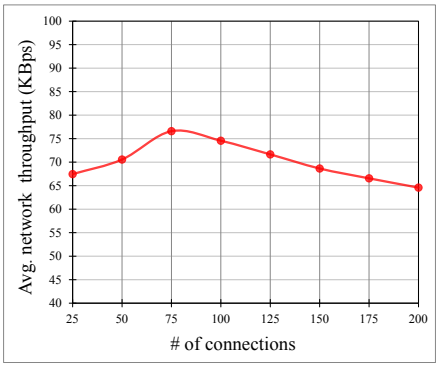
7.2 Network Performance Using GALS

In this section, we investigate CRN performance using GALS as the underlined DSA algorithm. For generating CRN environments and evaluating network performance, we followed the simulation settings that we discussed above. In this performance evaluation, we consider the following QoS parameters:

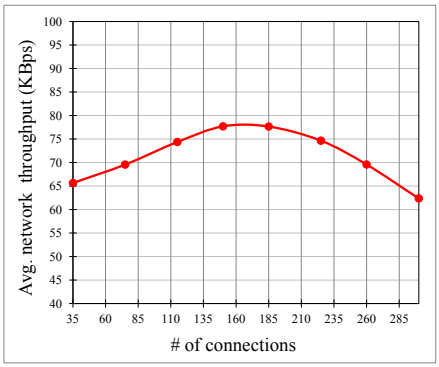
- **Network Throughput:** It refers to the amount of data transmitted across the network in a given time, usually measured in kilo-bytes per second (KBps). At each run, we calculated the average network throughput by averaging the individual data rates of all CR users for a given topology.



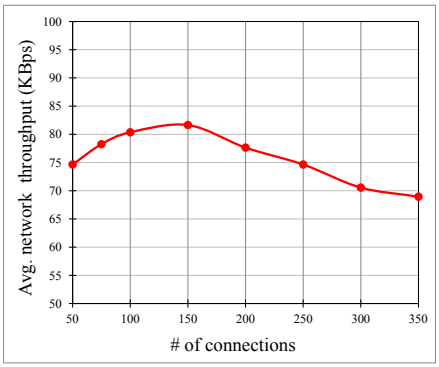
(a) Topology 1



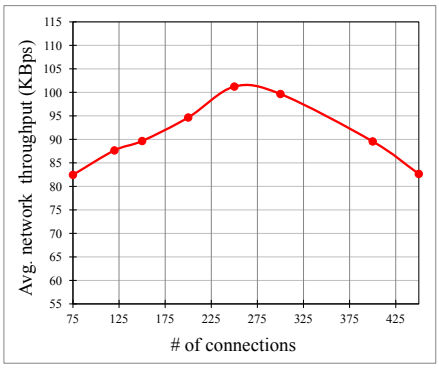
(b) Topology 2



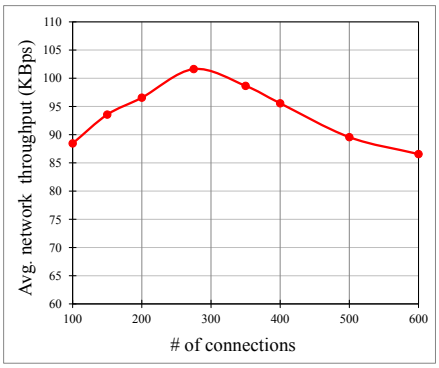
(c) Topology 3



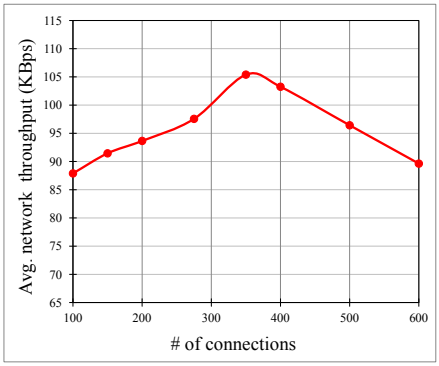
(d) Topology 4



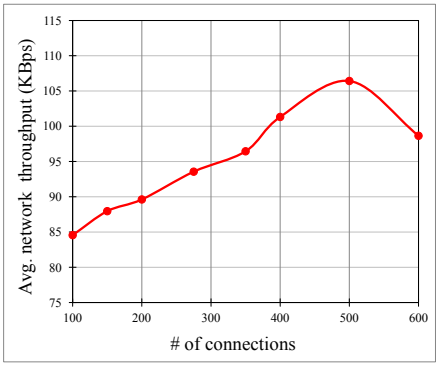
(e) Topology 5



(f) Topology 6

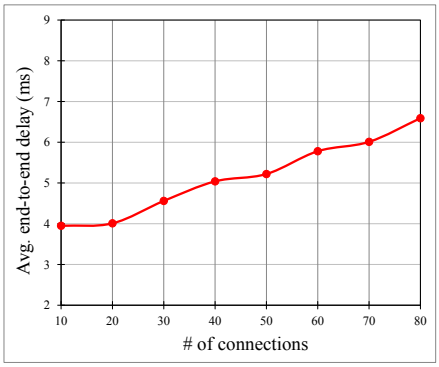


(g) Topology 7

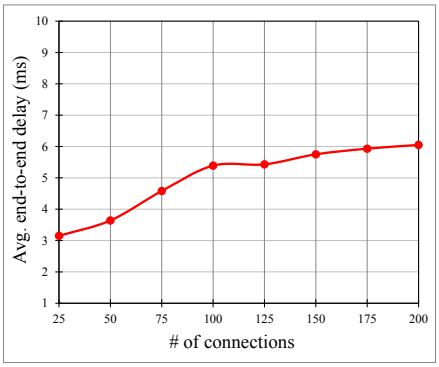


(h) Topology 8

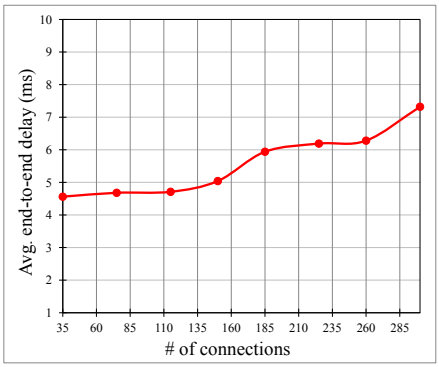
Figure 7.1: Average network throughput over different network topologies presented in Table 7.1



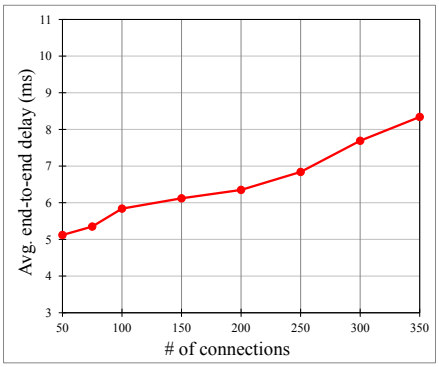
(a) Topology 1



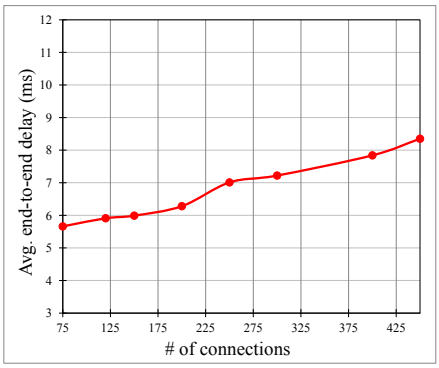
(b) Topology 2



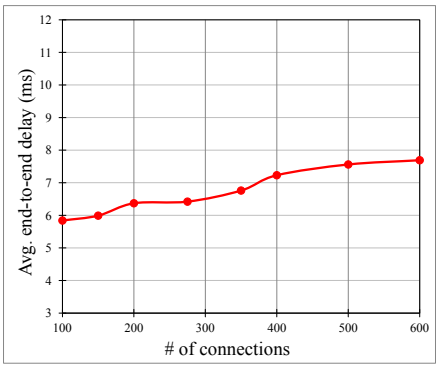
(c) Topology 3



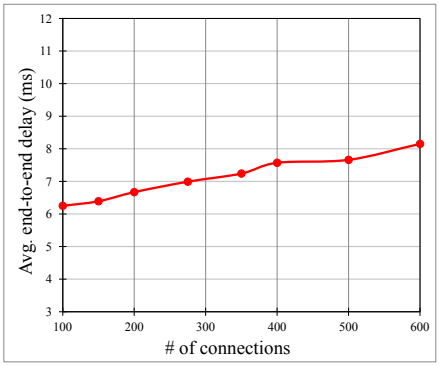
(d) Topology 4



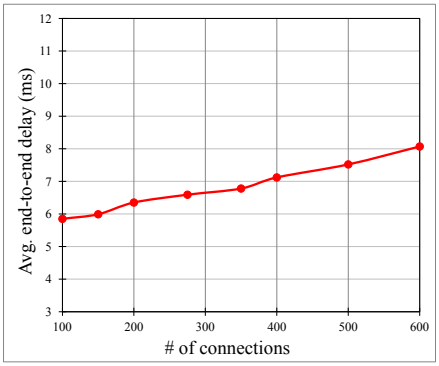
(e) Topology 5



(f) Topology 6

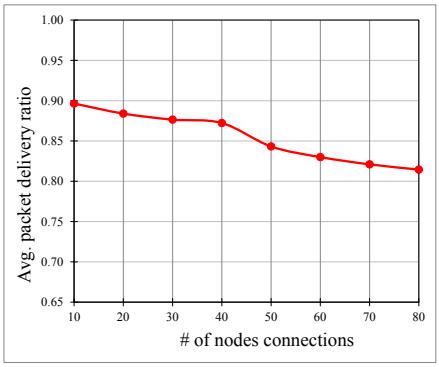


(g) Topology 7

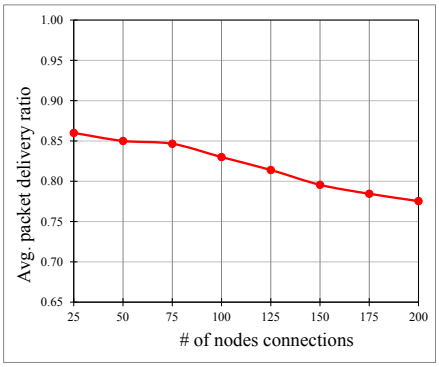


(h) Topology 8

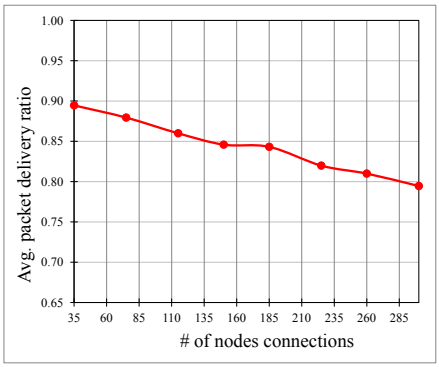
Figure 7.2: Average end-to-end delay over different network topologies presented in Table 7.1



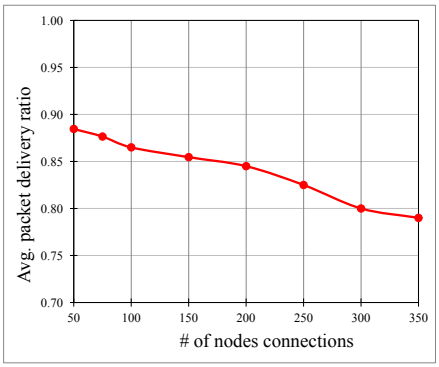
(a) Topology 1



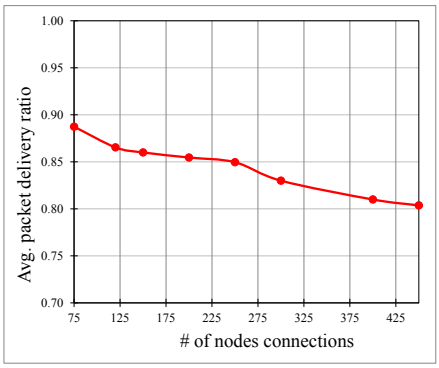
(b) Topology 2



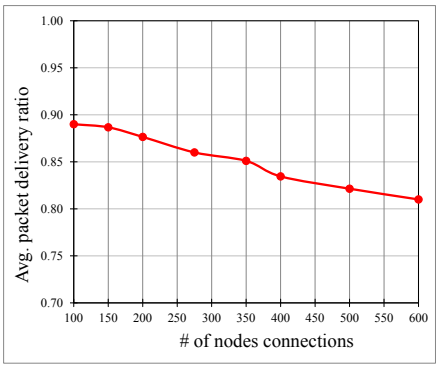
(c) Topology 3



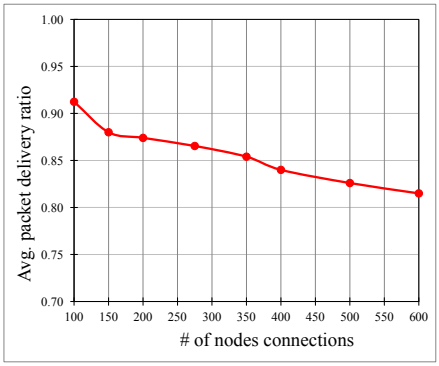
(d) Topology 4



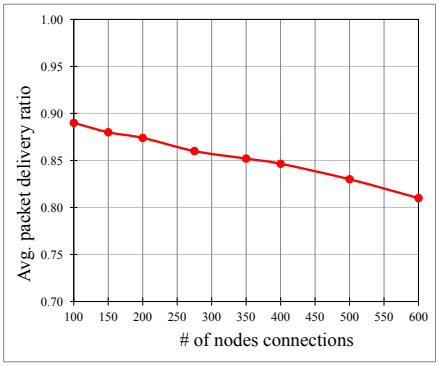
(e) Topology 5



(f) Topology 6

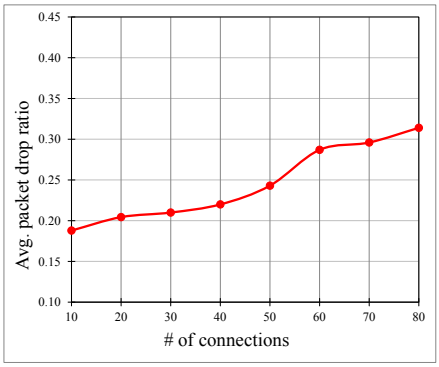


(g) Topology 7

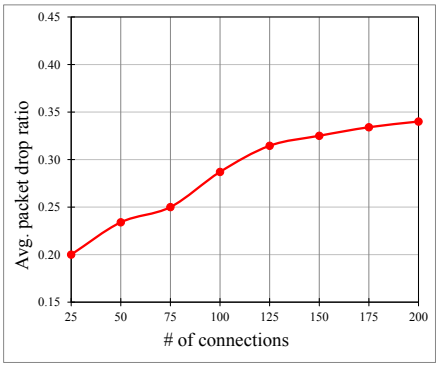


(h) Topology 8

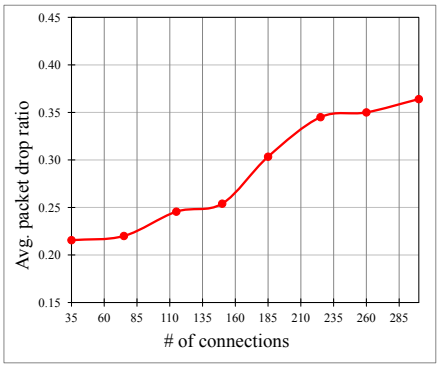
Figure 7.3: Average packet delivery ratio over different network topologies presented in Table 7.1



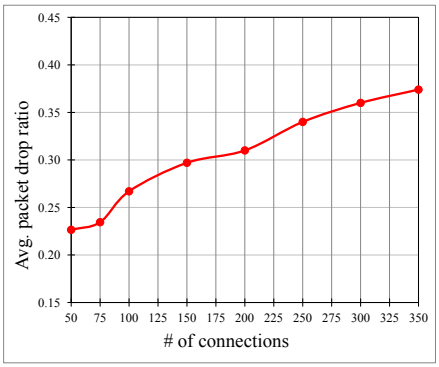
(a) Topology 1



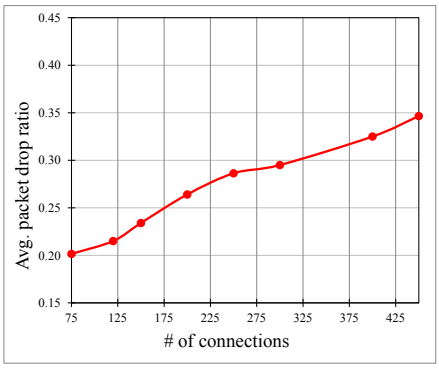
(b) Topology 2



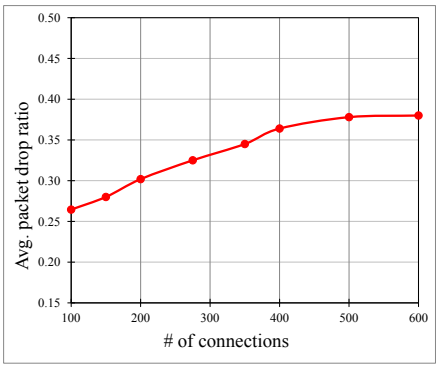
(c) Topology 3



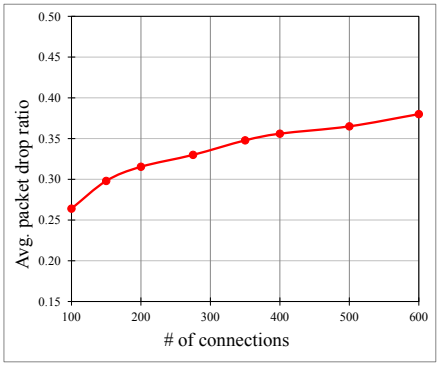
(d) Topology 4



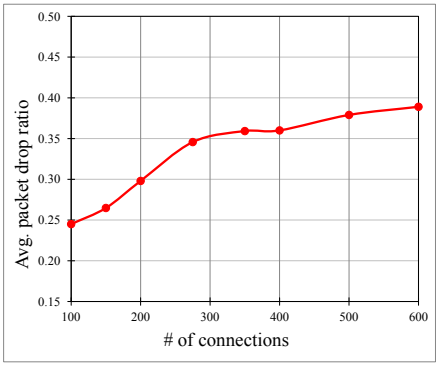
(e) Topology 5



(f) Topology 6



(g) Topology 7



(h) Topology 8

Figure 7.4: Average packet drop ratio over different network topologies presented in Table 7.1

We show the average network throughput over different network topologies in Figure 7.1. These illustrations follow that, as the number of connections increases up to a certain level, the network throughput increases. When the number of connections increases further, the network throughput degrades. Besides, the network usually demonstrate higher throughput when the number of connections per CR user is between 1 and 2.

- **End-to-end delay:** The time taken for a packet to be transmitted across the network from source to destination is referred to as end-to-end delay. It is usually measured in milliseconds (ms). In our simulation, we traced the sending and receiving times of each packet to find its end-to-end delay. Then, we averaged the end-to-end delays of all successful transmissions.

  We illustrate the average end-to-end delay of the network over different network topologies in Figure 7.2. This figure demonstrates that for all cases in general, the average end-to-end delay increases almost linearly with the increase of number of connections in the network.

- **Packet delivery ratio and packet drop ratio:** The ratio of number of packets that are successfully delivered to a destination compared to the number of packets that have been sent out by the sender, is referred to as packet delivery ratio. On the other hand, the ratio of the number of packets dropped by a sender to the total number of packets that it sent out, is referred to as packet drop ratio. Relation between packet delivery and drop ratio is not necessarily inverse (i.e., sum of them is not necessarily 1). This is because, one single packet can be dropped multiple times before its successful transmission.

  We present the average packet delivery ratio in Figure 7.3 and average packet drop ratio in Figure 7.4. These two QoS parameters also demonstrate almost linear change with an increase in number of connections in the network.

7.3 Performance Comparison with State-of-the-art Algorithms

Now, we evaluate and compare the performances of the different DSA techniques over varying network topologies. In the previous section, we discussed how we vary the number of CR users (n), number of channels (m), number of connections, and data rate to generate different network topologies. In this section, we discuss the state-of-the-art algorithms that we consider for performance comparison.

According to our discussion in Chapter 2, GALS is a distributive and cooperative DSA approach. Therefore, for performance comparison, we consider few state-of-the-art techniques that are both distributive and cooperative. The algorithms under comparison are Dynamic Conflict Graph (DCG) based DSA [14], Greedy Graph Coloring Algorithm (G-GCA) [15], Heuristic based Graph Coloring Algorithm (H-GCA) ([18], [21]), Game Theory (GT) based DSA [23], and Genetic Algorithm (GA) based DSA ([34], [35]) with best combination of genetic operators found in Chapter 4.

- **Dynamic Conflict Graph (DCG) Based DSA:** DSA algorithms based on network conflict graph [7] are the most common graph based DSA algorithms ([7], [14]). In such algorithms, we construct a network conflict graph that represents the interference between pairs of SUs. Dynamic conflict graphs are formed at each step of the DSA algorithm and take into account the aggregated interference effect. The SUs themselves form the sets of available channels and negotiate with their neighbors which channels to select in order to avoid interference between the links. All such approaches, according to our knowledge, consider network with SUs only. However, we incorporate the presence of PUs by adding a penalty term (similar to Equation 4.1, Chapter 4) to avoid interfering PUs. The details on the implementation procedure of the algorithm can be found in [14].

- **Graph Coloring Algorithm (GCA) Based DSA :** In graph coloring based approaches, we map the network to a graph, with CR users as vertices and interference between them as bidirectional edges. Then we deploy vertex or edge coloring of the graph, with m colors corresponding to m channels. If there is no m-coloring of the graph, then the coloring with the minimum conflicts is selected. GCA based DSA is implemented in many ways in the literature [7]. We use two distributive approaches. One performs the greedy coloring (G-GCA) and the other one uses heuristic based approximation for coloring (H-GCA). The greedy coloring (G-GCA) is most common in practice. However, due to NP-completeness of the graph coloring problem, heuristic based approximation coloring is used in literature ([18], [21]) to improve its performance. We perform the necessary adjustments and modifications that are required for implementing these algorithms in our single-radio multi-channel CRN, with consideration of both PUs. For details on the implementation procedure of GCA, we refer to [15], [18], and [21].

- **Game Theory (GT) Based DSA:** The basic model of game theory based DSA in CRNs is $G = \{n, S_i, U_i\}, i \in n$; where n is the number of CR users, which is perceived as the players. S_i

is the strategies of players, which is perceived as the actions. U_i is the set of utility function that the players associate with their strategies. Based on the communication pattern among players, utility function design, and overall the game formulation, there are a number of game theory based approaches ([22], [23], [55], [56]) in the literature which we already discussed previously. In our performance comparison, we consider a cooperative and distributive game formulation with interference based utility function. For the details of the utility function design and intuition to reach Nash Equilibrium, we refer to [56].

- **Genetic Algorithm (GA):** GA is the most widely used evolutionary algorithm for DSA ([34], [57]). We consider a GA based DSA as designed in [57]. That is, we choose the operators and adopt the parameter values for GA as discussed in [57]. Besides, we also compare performance of GA with the best operators and parameter values that we have found in Chapter 4. We name this setting as GA_{best}.

7.4 Simulation Results and Findings

Now, we present the performance comparison of GALS and the state-of-the-art algorithms over varied network topologies. We already mentioned the following QoS parameters for our evaluation: average network throughput, average end-to-end delay, average packet delivery ratio, and average packet drop ratio. In addition to these QoS parameters, we inspect fairness of the network as well. For comparing the fairness, we adopt two parameters: standard deviation of channel interference and standard deviation of per-node throughput, which we denote as δ_{CI} and δ_{PNT} respectively.

Here, in case of unfair DSA operation, few SUs will achieve much higher throughput, causing starvation to other SUs. Therefore, the values of per-node throughput will differ a lot, resulting in higher values of δ_{PNT}. Therefore, lower values of δ_{PNT} refer to fair DSA operation. Similarly, in case of unfair channel utilization, interference caused in different channels will vary a lot resulting in higher values of δ_{CI}. Therefore, lower values of δ_{CI} refer to fair DSA operations as well.

We present the first part of our simulation results in Figure 7.5. Here, we take the number of connections per CR user to be 1, $n : m$ to be $5 : 1$, and generate different network topologies by varying m from 5 to 30. In Figure 7.5(a) through Figure 7.5(f), we present the performance comparison in terms of average network throughput, average end-to-end delay, average packet delivery ratio, average packet drop ratio, δ_{CI} and δ_{PNT}, respectively.

Then, in terms of same QoS parameters, we present the performance evaluation with varied n in Figure 7.6. Here, we fix m to be 10, number of connections per CR user to be 1, and vary n from 20 to 200 to generate different network topologies.

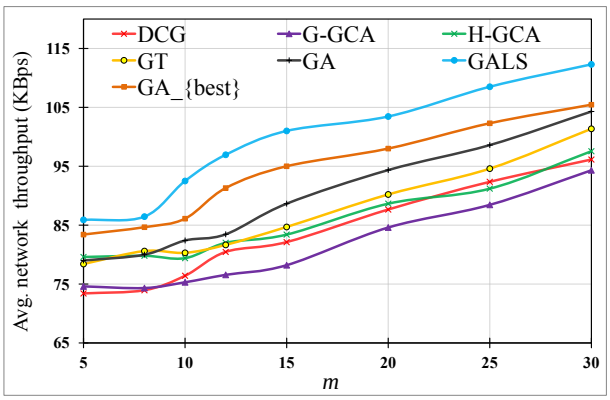
Subsequently, we present the performance evaluation with varied m in Figure 7.7. Here, we fix n to be 100, number of connections per CR user to be 1, and vary m from 4 to 32 to generate different network topologies.

Next to that, in Figure 7.8, we present the performance evaluation over the network topologies presented in Table 7.1. Here, we fix the number of connections per CR user to be 1. Then, in Figure 7.9, we present the performance comparison with varied number of connections per CR user. In this case, we consider a network topology with 100 CR users and 12 channels (i.e., topology 4 in Table 7.1), and vary the average number of connections per CR users from 1 to 6.

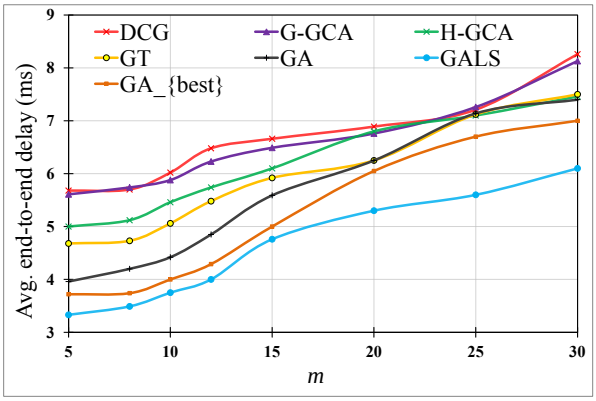
Finally, we present the performance evaluation with varied data rates in Figure 7.10. Here also, we consider a network topology with 100 CR users and 12 channels (i.e., topology 4 in Table 7.1). Besides, we fix the number of connections per CR user to be 1, and vary packets per second from 50 to 500 to investigate the performance.

For all these simulation cases mentioned above, we present the percentage of improvement using GALS compared to using other algorithms in Table 7.3 and Table 7.4. In Table 7.3, we illustrate the performance improvements of GALS in terms of average network throughput, average end-to-end delay, average packet delivery ratio, and average packet drop ratio. On the other hand, we present the performance improvements of GALS in terms of fairness (i.e., δ_{CI} and δ_{PNT}) in Table 7.4.

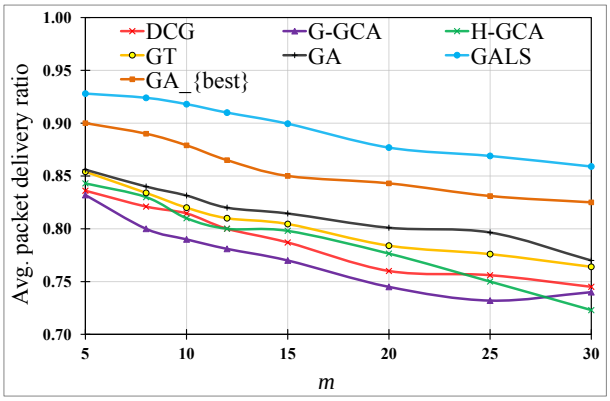
In Table 7.3 and Table 7.4, we also present the performance improvement by GA_{best} to demonstrate the effectiveness of our algorithmic design (i.e., chromosome representation, fitness function design, etc.) and to validate our primary experimental results, which we have presented in Chapter 4, for finding the best combination of operators and parameter values for GA-based DSA.



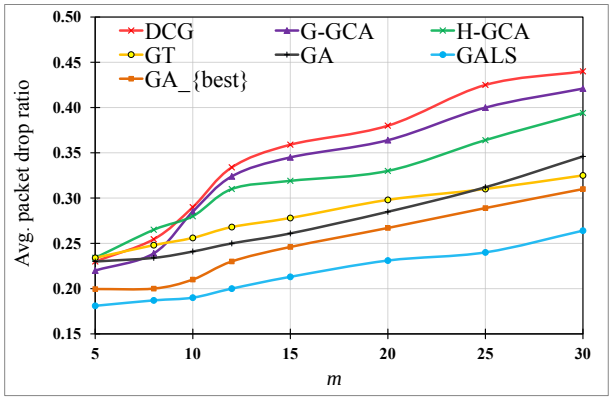
(a) Average network throughput



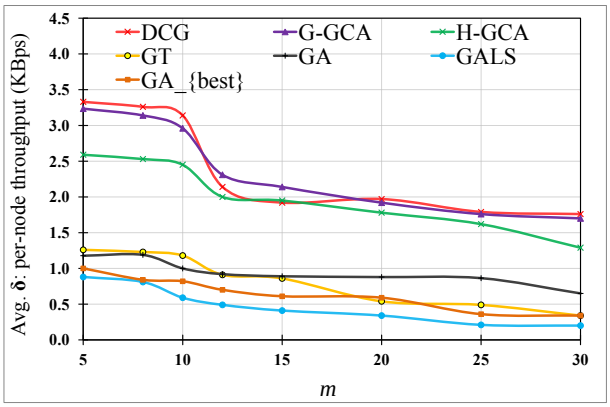
(b) Average end-to-end delay



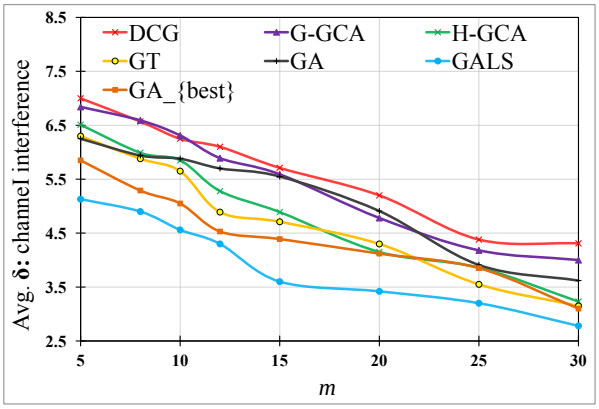
(c) Average packet delivery ratio



(d) Average packet drop ratio

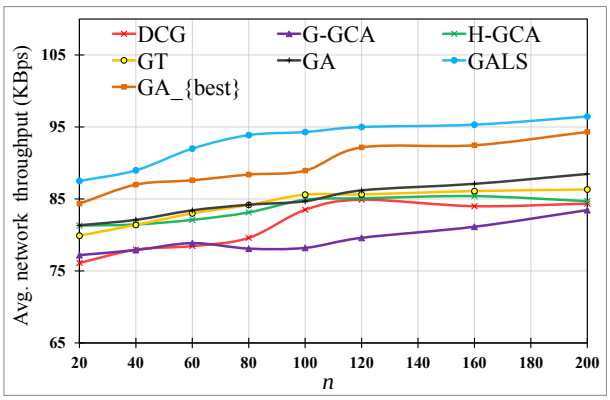


(e) Average δ_{PNT}

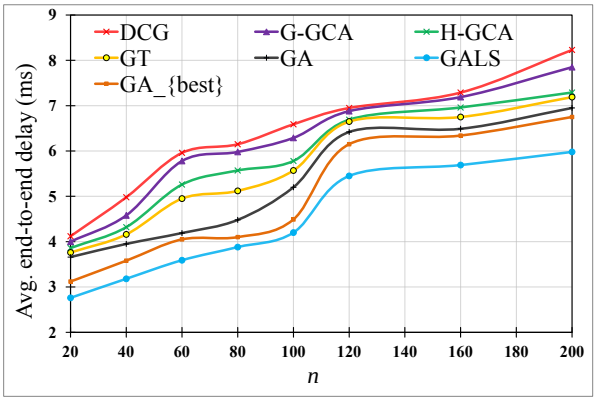


(f) Average δ_{CI}

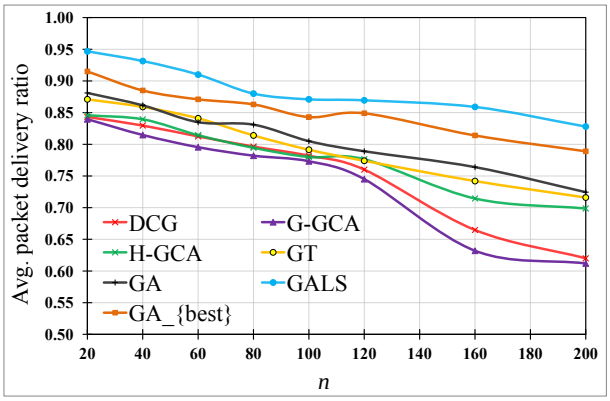
Figure 7.5: Performance comparison of DSA approaches in terms of various QoS parameters considering 5 CR users per channel



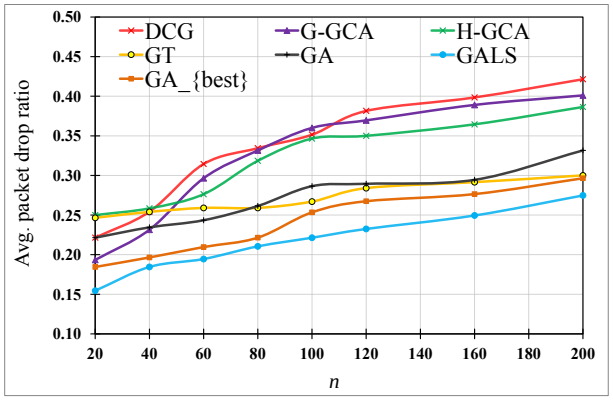
(a) Average network throughput



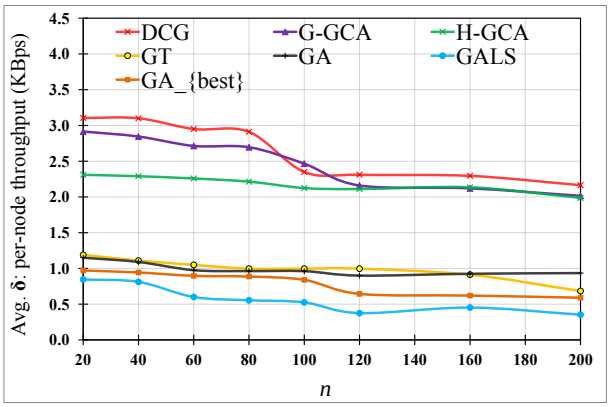
(b) Average end-to-end delay



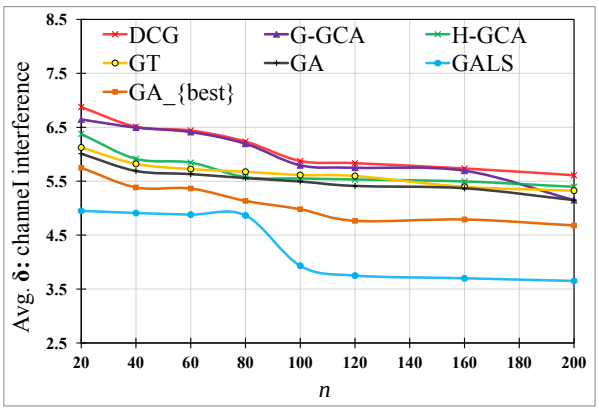
(c) Average packet delivery ratio



(d) Average packet drop ratio

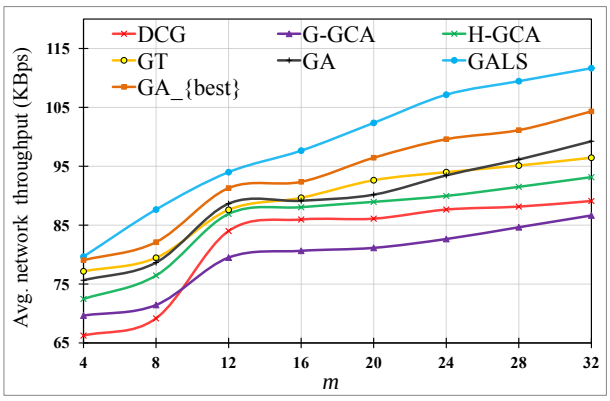


(e) Average δ_{PNT}

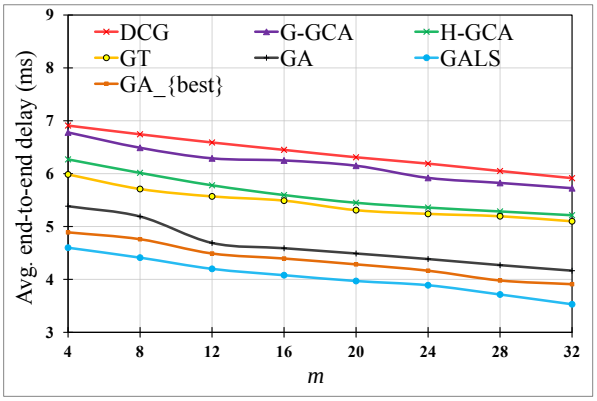


(f) Average δ_{CI}

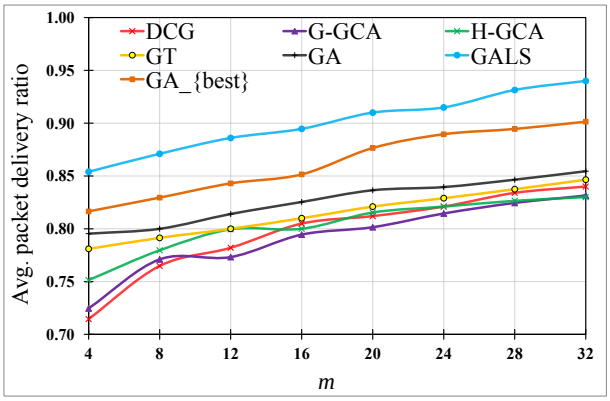
Figure 7.6: Performance comparison of DSA approaches in terms of various QoS parameters over varied number of CR users considering 10 channels



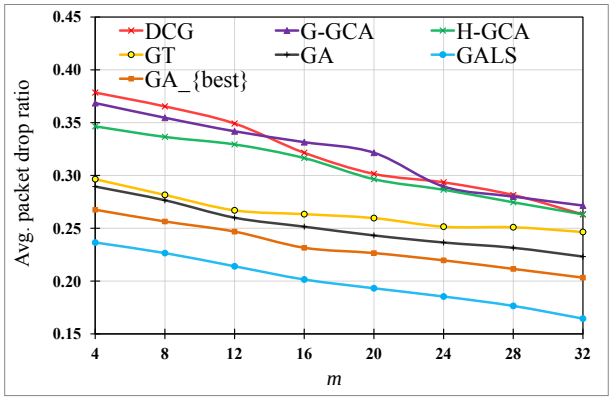
(a) Average network throughput



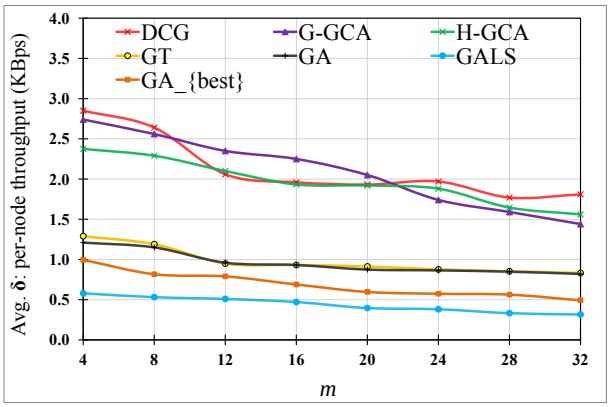
(b) Average end-to-end delay



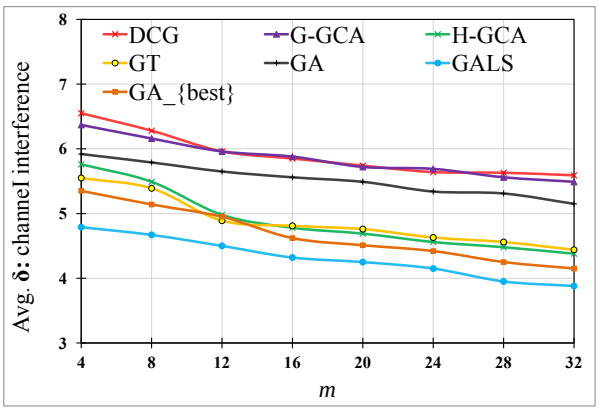
(c) Average packet delivery ratio



(d) Average packet drop ratio

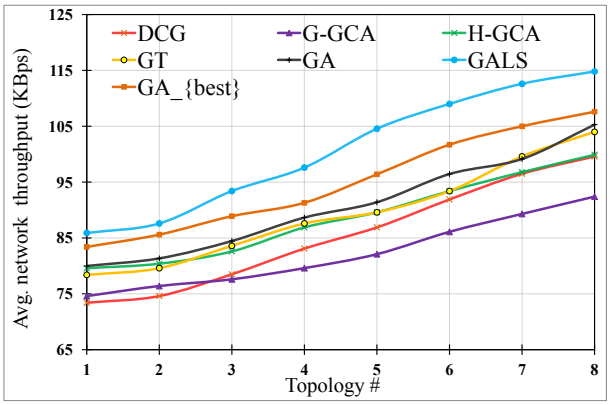


(e) Average δ_{PNT}

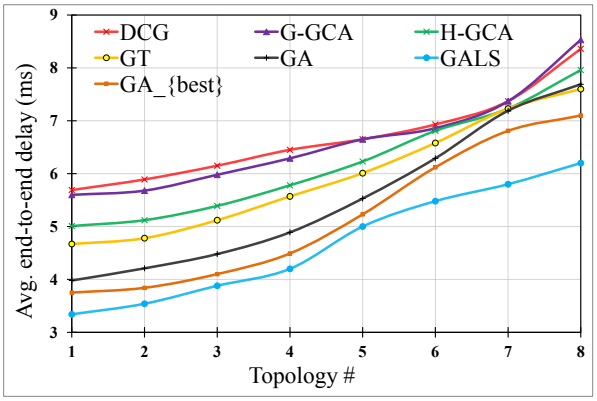


(f) Average δ_{CI}

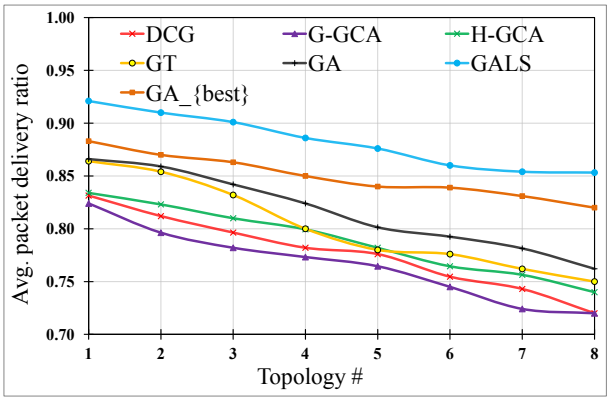
Figure 7.7: Performance comparison of DSA approaches in terms of various QoS parameters over varied number of channels considering 100 CR users



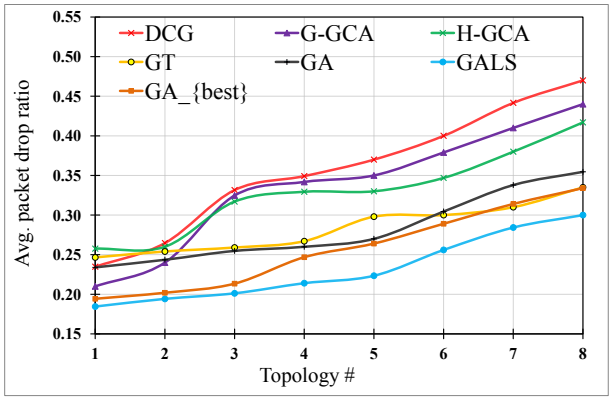
(a) Average network throughput



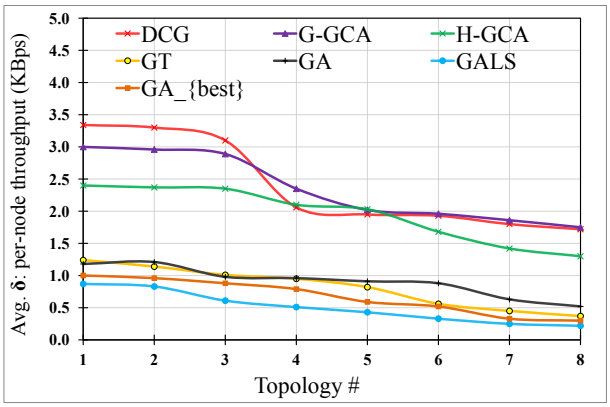
(b) Average end-to-end delay



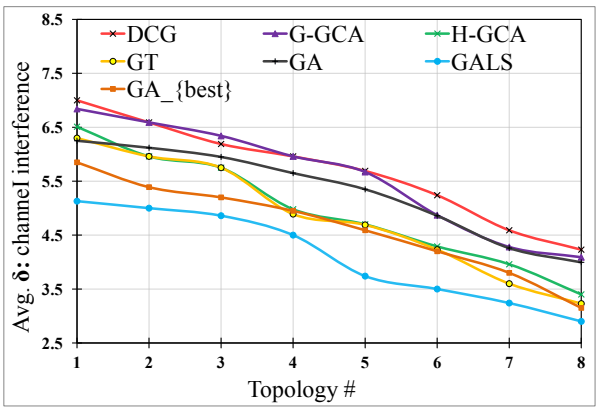
(c) Average packet delivery ratio



(d) Average packet drop ratio

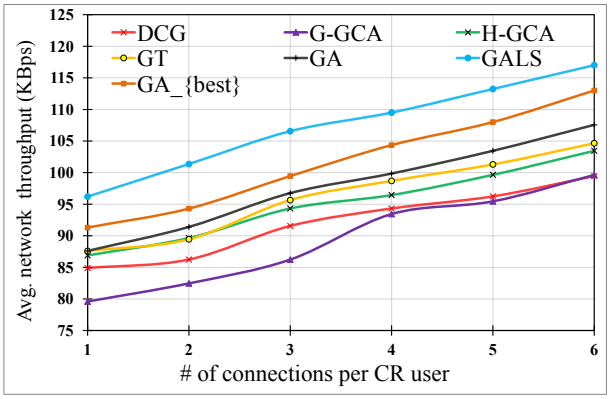


(e) Average δ_{PNT}

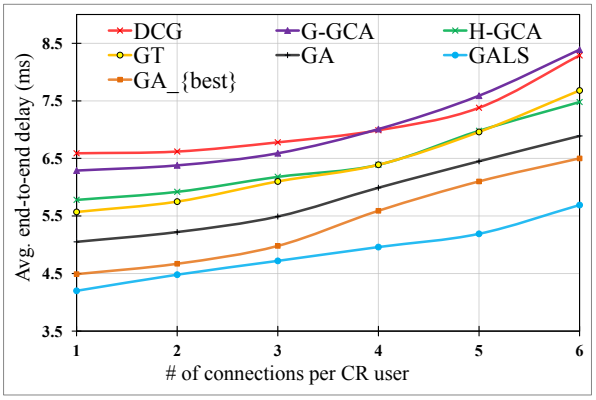


(f) Average δ_{CI}

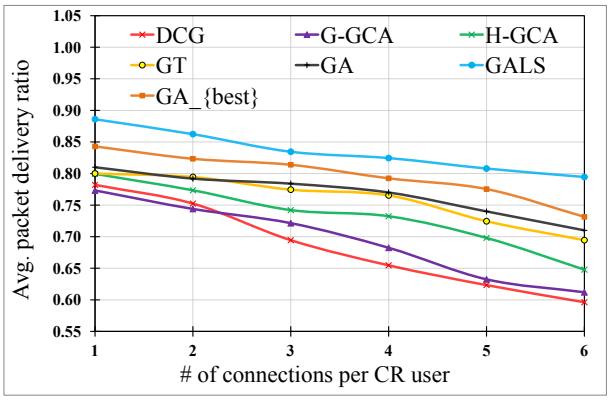
Figure 7.8: Performance comparison of DSA approaches in terms of various QoS parameters over varied network topologies presented in Table 7.1



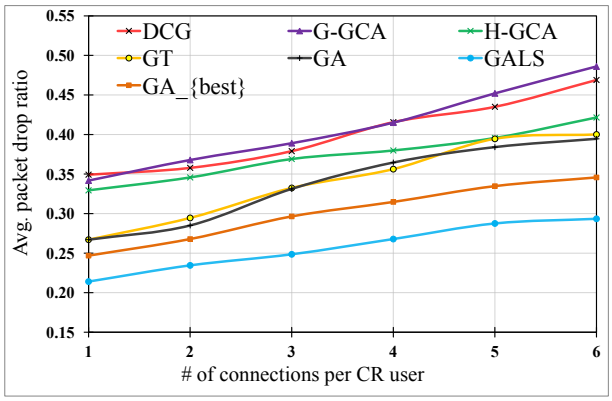
(a) Average network throughput



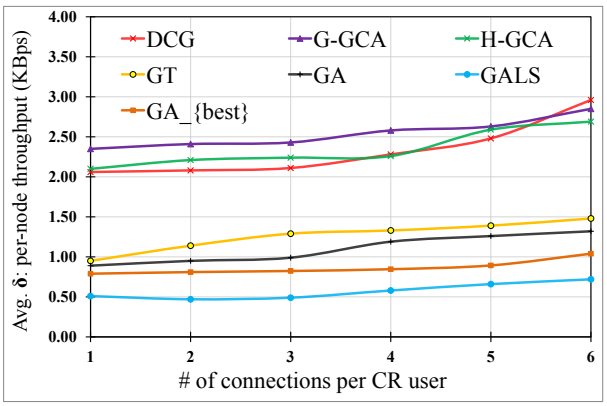
(b) Average end-to-end delay



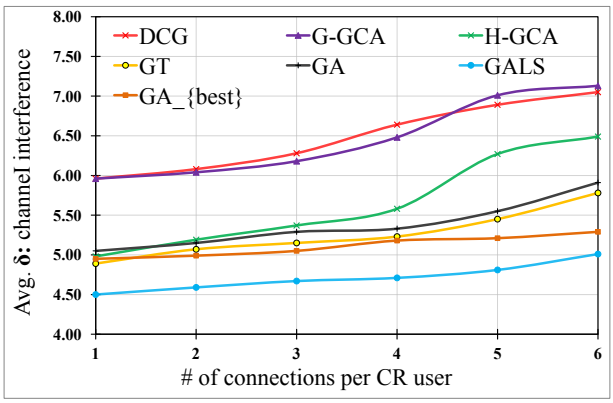
(c) Average packet delivery ratio



(d) Average packet drop ratio

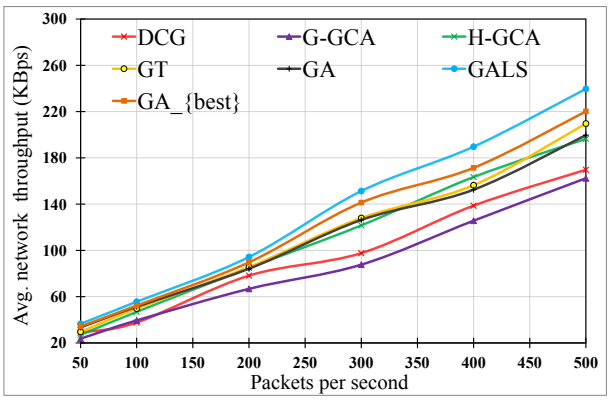


(e) Average δ_{PNT}

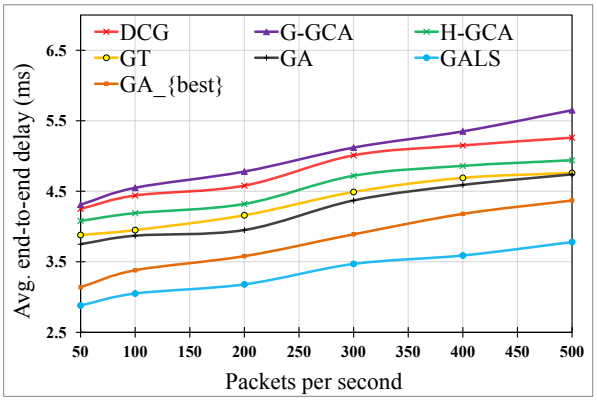


(f) Average δ_{CI}

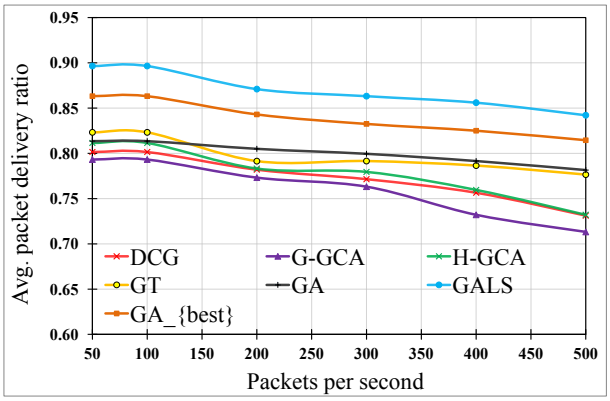
Figure 7.9: Performance comparison of DSA approaches in terms of various QoS parameters over varied number of connections per CR user considering topology 4 in Table 7.1



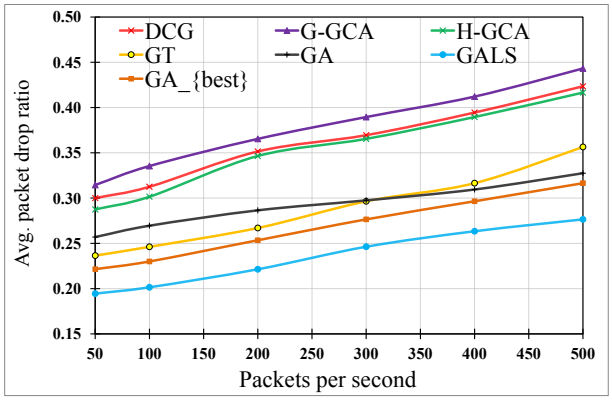
(a) Average network throughput



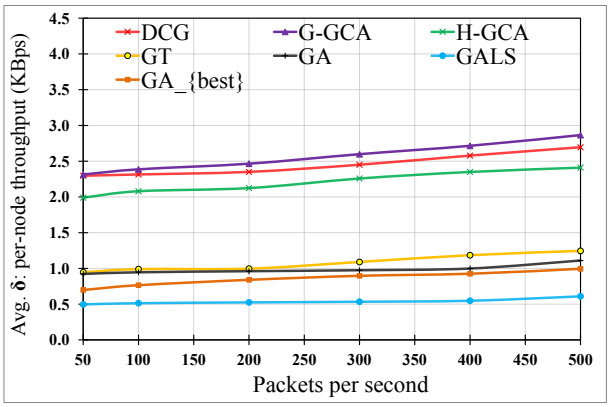
(b) Average end-to-end delay



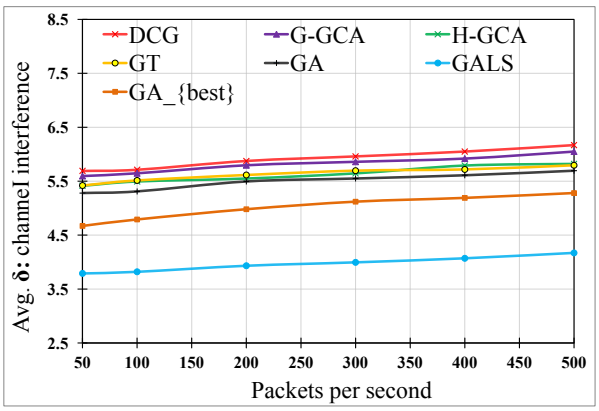
(c) Average packet delivery ratio



(d) Average packet drop ratio



(e) Average δ_{PNT}



(f) Average δ_{CI}

Figure 7.10: Performance comparison of DSA approaches in terms of various QoS parameters over varied data rates cconsidering topology 4 in Table 7.1

Table 7.3: Performance improvements by GA_{best} and GALS compared to DCG, G-GCA, H-GCA, GT, and GA

(a) Pertinent to the performance comparison presented in Figure 7.5

Algorithm in comparison	Network Throughput		Packet Delivery Ratio		Packet Drop Ratio		End-to-end-delay	
	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	13	19	9	14	28	38	23	31
G-GCA ([15], [18])	16	23	11	17	25	35	22	31
H-GCA ([18], [21])	10	16	9	14	22	32	17	26
GT [56]	8	14	7	9	12	24	13	22
GA [57]	5	11	6	12	10	21	7	17

(b) Pertinent to the performance comparison presented in Figure 7.6

Algorithm in comparison	Network Throughput		Packet Delivery Ratio		Packet Drop Ratio		End-to-end-delay	
	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	10	15	12	17	28	36	23	31
G-GCA ([15], [18])	13	18	14	19	26	34	21	29
H-GCA ([18], [21])	7	13	9	14	25	33	16	24
GT [56]	6	11	7	12	13	22	13	21
GA [57]	5	10	5	10	12	21	7	16

(c) Pertinent to the performance comparison presented in Figure 7.7

Algorithm in comparison	Network Throughput		Packet Delivery Ratio		Packet Drop Ratio		End-to-end-delay	
	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	14	21	8	13	27	37	30	37
G-GCA ([15], [18])	17	24	9	14	27	37	29	35
H-GCA ([18], [21])	9	15	8	13	24	35	22	28
GT [56]	5	11	6	11	12	25	20	26
GA [57]	5	11	5	10	7	21	6	13

Table 7.3: Performance improvements by GA_{best} and GALS compared to DCG, G-GCA, H-GCA, GT, and GA (continued)

(d) Pertinent to the performance comparison presented in Figure 7.8

Algorithm in comparison	Network Throughput		Packet Delivery Ratio		Packet Drop Ratio		End-to-end-delay	
	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	11	18	10	14	28	36	22	31
G-GCA ([15], [18])	16	22	11	16	24	32	22	31
H-GCA ([18], [21])	7	14	8	12	19	31	16	25
GT [56]	6	13	7	11	9	19	12	22
GA [57]	4	11	5	15	9	18	7	17

(e) Pertinent to the performance comparison presented in Figure 7.9

Algorithm in comparison	Network Throughput		Packet Delivery Ratio		Packet Drop Ratio		End-to-end-delay	
	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	10	17	16	22	24	36	24	31
G-GCA ([15], [18])	13	20	14	21	26	37	23	31
H-GCA ([18], [21])	7	13	9	14	19	31	17	25
GT [56]	6	12	5	11	12	25	16	24
GA [57]	4	11	5	15	11	24	8	17

(f) Pertinent to the performance comparison presented in Figure 7.10

Algorithm in comparison	Network Throughput		Packet Delivery Ratio		Packet Drop Ratio		End-to-end-delay	
	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	28	40	9	15	26	35	21	31
G-GCA ([15], [18])	39	50	11	15	29	39	24	33
H-GCA ([18], [21])	10	20	8	12	24	34	17	27
GT [56]	8	17	7	11	7	19	13	24
GA [57]	10	19	5	10	9	20	11	22

Table 7.4: Performance improvement (%) by GA_{best} and GALS in terms of fairness

(a) Pertinent to the performance comparison presented in Figure 7.5

Algorithm in comparison	δ_{PNT}		δ_{CI}	
	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	73	80	20	30
G-GCA ([15], [18])	72	80	18	28
H-GCA ([18], [21])	66	76	9	20
GT [56]	22	43	6	17
GA [57]	30	49	13	24

(b) Pertinent to the performance comparison presented in Figure 7.6

Algorithm in comparison	δ_{PNT}		δ_{CI}	
	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	69	79	17	30
G-GCA ([15], [18])	67	78	16	28
H-GCA ([18], [21])	63	75	11	25
GT [56]	20	44	10	24
GA [57]	19	44	8	22

(c) Pertinent to the performance comparison presented in Figure 7.7

Algorithm in comparison	δ_{PNT}		δ_{CI}	
	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	67	80	21	27
G-GCA ([15], [18])	67	79	20	27
H-GCA ([18], [21])	65	78	5	12
GT [56]	30	55	5	13
GA [57]	28	55	15	23

(d) Pertinent to the performance comparison presented in Figure 7.8

Algorithm in comparison	δ_{PNT}		δ_{CI}	
	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	71	79	19	28
G-GCA ([15], [18])	70	79	17	27
H-GCA ([18], [21])	64	74	7	18
GT [56]	18	38	5	15
GA [57]	26	44	13	22

Table 7.4: Performance improvement (%) by GA_{best} and GALS in terms of fairness (continued)

(e) Pertinent to the performance comparison presented in Figure 7.9

Algorithm in comparison	δ_{PNT}		δ_{CI}	
	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	61	75	21	27
G-GCA ([15], [18])	65	78	21	28
H-GCA ([18], [21])	63	77	10	17
GT [56]	31	55	5	11
GA [57]	21	49	6	14

(f) Pertinent to the performance comparison presented in Figure 7.10

Algorithm in comparison	δ_{PNT}		δ_{CI}	
	GA_{best}	GALS	GA_{best}	GALS
DCG [14]	65	79	15	33
G-GCA ([15], [18])	66	79	14	32
H-GCA ([18], [21])	61	77	11	30
GT [56]	21	50	11	30
GA [57]	13	46	9	28

According to the illustrations in Figure 7.5 through Figure 7.10, Table 7.3, and Table 7.4, we summarize the findings of our simulation results below:

- In general, GA and GT perform slightly better than G-GCA and H-GCA, whereas, GA_{best} performs better than GA and GT. On the other hand, GALS always achieves significantly better performance than all these algorithms under comparison.

- GALS achieves marginally better performance compared to GA_{best} in smaller networks (i.e., with relatively less number of CR users). However, in larger networks, GALS achieves much better performance compared to GA_{best}. In addition to improved performance, GALS demonstrates more consistent performance with varying network scenarios compared to the state-of-the-art algorithms.

- According to Table 7.3, GALS achieves consistent performance improvement compared to the state-of-the-art DSA algorithms over all the simulation cases. Overall, GALS achieve $10-50\%$, $10-22\%$, $18-39\%$, and $13-37\%$ improvement on average network throughput, packet delivery ratio, packet drop ratio, and end-to-end delay, respectively, compared to the state-of-the-art DSA algorithms.

- The job of DSA becomes more challenging with an increasing number of connections per CR users, which is evident from the simulation results in Figure 7.9. Here, network performance of CRNs, particularly in terms of average end-to-end-delay, packet drop ratio, and fairness, degrade significantly. Here, as shown in Table 7.3(b), GALS achieves up to 20%, 22%, 38%, and 31% improvement on average network throughput, packet delivery ratio, packet drop ratio, and end-to-end delay, respectively, compared to the state-of-the-art DSA algorithms.

- Although performance improvement of GALS in terms of average network throughput, packet delivery ratio, packet drop ratio, and end-to-end delay is quite good, its performance in terms of fairness is rather significant. As Figure 7.5 through 7.10 suggest, GALS achieves much lower values of δ_{PNT} and δ_{CI}, resulting in more fair DSA operation in CRNs. According to Table 7.4, GALS achieves up to 80% and 33% improvement on δ_{PNT} and δ_{CI}, respectively. Therefore, we find that GALS achieve improved per-node throughput and fairness in channel utilization compared to other state-of-the-art approaches.

- In Chapter 4, we have presented our chromosome representation, designed our fitness function, and performed thorough experimentations to find out the best operators and parameter values for GA based DSA. Significant performance improvement of GA_{best} compared to G-GCA, H-GCA, GT, and GA, presented in Table 7.3 and Table 7.4, exhibits effectiveness of our algorithmic design and thus validates our primary experimental results.

- In all the simulation cases, GALS achieves better performance than GA_{best}. This improved performance further demonstrates effectiveness of our novel genetic operators.

We have already presented the improved performance of GALS in terms of convergence in Table 5.1 (Chapter 5). With all these results and observations mentioned above, we conclude that our hybrid algorithm GALS performs significantly better than other state-of-the-art algorithms. In particular, fairness is known to be a difficult parameter to improve in distributed CRNs [7], as many state-of-the-art algorithms are known to perform worse in terms of this parameter ([7], [40]). GALS demonstrates remarkable performance improvement on fairness of the network (both in terms of per-node throughput and channel utilization). Therefore, GALS is a highly scalable and efficient DSA algorithm for multi-channel single-radio CRNs, that ensures network fairness in addition to achieving better performance in terms of other QoS parameters as well.

Chapter 8

Conclusion

Various stochastic and classical approaches perform DSA in CRNs, however, these approaches exhibit different pros and cons while operating in a network. We propose to exploit a synergy between the two types of approaches to overcome those limitations. Consequently, in this paper, we proposed a hybrid technique (GALS) for multi-channel single-radio CRNs.

In this paper, first, we thoroughly investigate the performance of traditional genetic operators, which eventually leads us to designing novel hybrid genetic operators. These novel hybrid genetic operators strike a delicate balance between classical and stochastic searching. We evaluate the performance of our proposed hybrid technique using discrete event simulator. Simulation results demonstrate significant performance improvement in terms of various performance metrics using our approach over state-of-the-art approaches. Specially the performance improvement in terms of fairness, a challenging and well-researched metric for distributive CRNs, using our proposed approach is significant (up to 79%).

In our simulation, we use CRCN simulator, which is based on `ns-2`. Before using, we perform several modifications that are necessary to mimic real CRN environment. These modifications can be exploited in future research studies pertinent for implementing and testing new DSA algorithms.

We plan to extend our work for multi-radio CRNs in future. Our plan covers exploring energy considerations in CRNs. Finally, design of efficient and scalable DSA algorithms for CRNs having mobile users (random movements as in MANETs and constrained movements as in VANETs) is another research topic we want to explore in future.

Bibliography

[1] V. A. Siris, E. Z. Tragos, and N. E. Petroulakis, "Experiences with a metropolitan multiradio wireless mesh network: design, performance, and application," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 128–136, 2012.

[2] B. Wang and K. R. Liu, "Advances in cognitive radio networks: A survey," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 1, pp. 5–23, 2011.

[3] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey," *Computer Networks*, vol. 50, no. 13, pp. 2127–2159, 2006.

[4] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "A survey on spectrum management in cognitive radio networks," *Communications Magazine, IEEE*, vol. 46, no. 4, pp. 40–48, 2008.

[5] "Cognitive-networks." `http://howdoesinternetwork.com/2012/cognitive-networks`. [Online; last accessed January-2015].

[6] "Jing Zhao's research." `http://www.cse.psu.edu/~juz139/research.html`. [Online; last accessed January-2015].

[7] E. Z. Tragos, S. Zeadally, A. G. Fragkiadakis, and V. A. Siris, "Spectrum assignment in cognitive radio networks: A comprehensive survey.," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1108–1135, 2013.

[8] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 2, pp. 201–220, 2005.

[9] "Nokia Research Center." `http://research.nokia.com/cognitive_radio`. [Online; last accessed January-2015].

[10] J. C. Li, W. Zhang, A. Nosratinia, and J. Yuan, "Sharp: spectrum harvesting with arq retransmission and probing in cognitive radio," *Communications, IEEE Transactions on*, vol. 61, no. 3, pp. 951–960, 2013.

[11] M. Pan, C. Zhang, P. Li, and Y. Fang, "Spectrum harvesting and sharing in multi-hop crns under uncertain spectrum supply," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 2, pp. 369–378, 2012.

[12] B. A. Fette, "Software-defined radio," Dec. 2 2004. US Patent 20,040,242,261.

[13] A. Plummer and S. Biswas, "Distributed spectrum assignment for cognitive networks with heterogeneous spectrum opportunities," *Wireless Communications and Mobile Computing*, vol. 11, no. 9, pp. 1239–1253, 2011.

[14] A. T. Hoang and Y.-C. Liang, "Maximizing spectrum utilization of cognitive radio networks using channel allocation and power control," in *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pp. 1–5, IEEE, 2006.

[15] X. Su, C. Yuan, and S. Shen, "A new mechanism of dynamic spectrum allocation in the cognitive network," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*, pp. 1–4, IEEE, 2009.

[16] A. T. Hoang and Y.-C. Liang, "Downlink channel assignment and power control for cognitive radio networks," *Wireless Communications, IEEE Transactions on*, vol. 7, no. 8, pp. 3106–3117, 2008.

[17] M. Bkassiny and S. K. Jayaweera, "Optimal channel and power allocation for secondary users in cooperative cognitive radio networks," in *Mobile Lightweight Wireless Systems*, pp. 180–191, Springer, 2010.

[18] A. Sampath, L. Yang, L. Cao, H. Zheng, and B. Y. Zhao, "High throughput spectrum-aware routing for cognitive radio networks," *Proc. of IEEE Crowncom*, 2008.

[19] M. Bkassiny and S. K. Jayaweera, "Optimal channel and power allocation for secondary users in cooperative cognitive radio networks," in *Mobile Lightweight Wireless Systems*, pp. 180–191, Springer, 2010.

[20] G. Liu, L. Zhou, K. Xiao, B. Yu, G. Zhou, B. Wang, and X. Zhu, "Receiver-centric channel assignment model and algorithm in cognitive radio network," in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pp. 1–4, IEEE, 2008.

[21] L. Yu, C. Liu, Z. Liu, and W. Hu, "Heuristic spectrum assignment algorithm in distributed cognitive networks," in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pp. 1–5, IEEE, 2010.

[22] L. Chen, S. Iellamo, M. Coupechoux, and P. Godlewski, "An auction framework for spectrum allocation with interference constraint in cognitive radio networks," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[23] Y.-B. Li, R. Yang, and F. Ye, "Non-cooperative spectrum allocation based on game theory in cognitive radio networks," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, pp. 1134–1137, IEEE, 2010.

[24] P. Klemperer, "Auction theory: A guide to the literature," *Journal of economic surveys*, vol. 13, no. 3, pp. 227–286, 1999.

[25] G. S. Kasbekar and S. Sarkar, "Spectrum auction framework for access allocation in cognitive radio networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 6, pp. 1841–1854, 2010.

[26] N. Nie, C. Comaniciu, and P. Agrawal, "A game theoretic approach to interference management in cognitive networks," in *Wireless Communications*, pp. 199–219, Springer, 2007.

[27] N. Nie and C. Comaniciu, "Adaptive channel allocation spectrum etiquette for cognitive radio networks," *Mobile networks and applications*, vol. 11, no. 6, pp. 779–797, 2006.

[28] A. Ahmed, M. M. Hassan, O. Sohaib, W. Hussain, and M. Q. Khan, "An agent based architecture for cognitive spectrum management.," *Australian Journal of Basic & Applied Sciences*, vol. 5, no. 12, 2011.

[29] C. Wu, K. Chowdhury, M. Di Felice, and W. Meleis, "Spectrum management of cognitive radio using multi-agent reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry track*, pp. 1705–1712, International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[30] H. Wang, J. Ren, and T. Li, "Resource allocation with load balancing for cognitive radio networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1–5, IEEE, 2010.

[31] Z. Wenzhu and L. Xuchen, "Centralized dynamic spectrum allocation in cognitive radio networks based on fuzzy logic and q-learning," *China Communications*, vol. 8, no. 7, pp. 46–54, 2011.

[32] H. G. Sandalidis and P. Stavroulakis, "Heuristics for solving fixed-channel assignment problems," *Handbook of wireless networks and mobile computing*, p. 51, 2002.

[33] S. Li, T. H. Luan, and X. Shen, "Channel allocation for smooth video delivery over cognitive radio networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1–5, IEEE, 2010.

[34] Z. Zhao, Z. Peng, S. Zheng, and J. Shang, "Cognitive radio spectrum allocation using evolutionary algorithms," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 9, pp. 4421–4425, 2009.

[35] F. Ye, R. Yang, and Y. Li, "Genetic algorithm based spectrum assignment model in cognitive radio networks," in *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on*, pp. 1–4, IEEE, 2010.

[36] T. Chen, H. Zhang, M. D. Katz, and Z. Zhou, "Swarm intelligence based dynamic control channel assignment in cogmesh," in *Communications Workshops, 2008. ICC Workshops' 08. IEEE International Conference on*, pp. 123–128, IEEE, 2008.

[37] H. Salehinejad, S. Talebi, and F. Pouladi, "A metaheuristic approach to spectrum assignment for opportunistic spectrum access," in *Telecommunications (ICT), 2010 IEEE 17th International Conference on*, pp. 234–238, IEEE, 2010.

[38] D. Jin, D. He, D. Liu, and C. Baquero, "Genetic algorithm with local search for community mining in complex networks," in *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, vol. 1, pp. 105–112, IEEE, 2010.

[39] "CRCN Simulator." `http://faculty.uml.edu/Tricia_Chigan/Research/CRCN_Simulator.htm`. [Online; last accessed January-2015].

[40] S.-S. Byun, I. Balasingham, and X. Liang, "Dynamic spectrum allocation in wireless cognitive sensor networks: Improving fairness and energy efficiency," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pp. 1–5, IEEE, 2008.

[41] F. E. D. No, "03-322," *Notice of proposed rule making and order*, 2003.

[42] G. Alnwaimi, K. Arshad, and K. Moessner, "Dynamic spectrum allocation algorithm with interference management in co-existing networks," *Communications Letters, IEEE*, vol. 15, no. 9, pp. 932–934, 2011.

[43] L. Ding, T. Melodia, S. N. Batalama, J. D. Matyjas, and M. J. Medley, "Cross-layer routing and dynamic spectrum allocation in cognitive radio ad hoc networks," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 4, pp. 1969–1979, 2010.

[44] X. Li and S. A. R. Zekavat, "Distributed channel assignment in cognitive radio networks," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, pp. 989–993, ACM, 2009.

[45] Q. Zhao and B. M. Sadler, "A survey of dynamic spectrum access," *Signal Processing Magazine, IEEE*, vol. 24, no. 3, pp. 79–89, 2007.

[46] C. Xin, L. Ma, and C.-C. Shen, "A path-centric channel assignment framework for cognitive radio wireless networks," *Mobile Networks and Applications*, vol. 13, no. 5, pp. 463–476, 2008.

[47] E. A. Silver, "An overview of heuristic solution methods," *Journal of the operational research society*, vol. 55, no. 9, pp. 936–956, 2004.

[48] B. Salameh and H. Ahmad, "Throughput-oriented channel assignment for opportunistic spectrum access networks," *Mathematical and Computer Modelling*, vol. 53, no. 11, pp. 2108–2118, 2011.

[49] P. Kaur, M. Uddin, and A. Khosla, "Adaptive bandwidth allocation scheme for cognitive radios.," *Int. J. Adv. Comp. Techn.*, vol. 2, no. 2, pp. 35–41, 2010.

[50] "Modifications in the basic CRCN simulator." `https://sites.google.com/site/xahidbuffon/shared/DSA_CRCN`. [Online; last accessed January-2015].

[51] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Urbana*, vol. 51, pp. 61801–2996, 1991.

[52] M. R. Noraini and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving tsp," 2011.

[53] D. Beasley, R. Martin, and D. Bull, "An overview of genetic algorithms: Part 1. fundamentals," *University computing*, vol. 15, pp. 58–58, 1993.

[54] F. Herrera, M. Lozano, and A. M. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study," *International Journal of Intelligent Systems*, vol. 18, no. 3, pp. 309–338, 2003.

[55] B. Wang, Y. Wu, and K. Liu, "Game theory for cognitive radio networks: An overview," *Computer networks*, vol. 54, no. 14, pp. 2537–2561, 2010.

[56] M. Maskery, V. Krishnamurthy, and Q. Zhao, "Decentralized dynamic spectrum access for cognitive radios: cooperative design of a non-cooperative game," *Communications, IEEE Transactions on*, vol. 57, no. 2, pp. 459–469, 2009.

[57] F. Ye, R. Yang, and Y. Li, "Genetic spectrum assignment model with constraints in cognitive radio networks," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 3, no. 4, p. 39, 2011.