Dynamic Bandwidth Allocation in MIMO 5G Networks

Nikolaos Prodromos
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: up1072549@upnet.gr

Damianos Diasakos
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: up1084632@upnet.gr

Vasileios Kokkinos
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: kokkinos@cti.gr

Apostolos Gkamas
Department of Chemistry
University of Ioannina
Ioannina, Greece
Email: gkamas@uoi.gr

Christos Bouras
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: bouras@upatras.gr

Philippos Pouyioutas
Computer Science Department
University of Nicosia
Nicosia, Cyprus
Email: pouyioutas.p@unic.ac.cy

***Abstract*— The advent of 5G technology has ushered in a new era of wireless communication, characterized by its promise of high data rates, low latency, and enhanced connectivity. In this context, Multiple-Input Multiple-Output (MIMO) systems have emerged as a key enabler, leveraging advanced antenna arrays to simultaneously serve multiple users with increased spectral efficiency. This paper investigates the dynamic resource allocation problem in a MIMO 5G environment, where each user possesses distinct bandwidth requirements. The focus is on optimizing user allocation while considering the limited bandwidth and user capacity of base stations. By harnessing the power of deep learning techniques, the proposed solution aims to efficiently manage the allocation of users to base station antennas, thereby maximizing overall network performance while accommodating heterogeneous user demands.**

Keywords*— *5G Networks, User Allocation, Bandwidth Allocation, Heterogeneous Networks

I. Introduction

The relentless growth in wireless data consumption, coupled with the proliferation of Internet of Things (IoT) devices, has spurred the development of 5G technology to cater to the escalating demands of diverse applications. Multiple-Input Multiple-Output (MIMO) systems, which employ multiple antennas at both the transmitter and receiver ends, have proven instrumental in realizing the ambitious goals of 5G networks, such as high data rates and massive device connectivity [1]. Building upon the successes of traditional MIMO, MIMO systems integrate techniques to unravel complex spatial correlations and channel behaviours, enabling the exploitation of multiuser diversity and achieving enhanced spectral efficiency.

In MIMO 5G environments, optimizing resource allocation is crucial due to diverse and dynamic bandwidth demands across various applications. With finite bandwidth and user capacity per base station, efficient management of user-to-antenna allocation is essential for network performance. This paper addresses the complex challenge of allocating users in MIMO 5G environments with varying bandwidth needs to enhance quality of service (QoS). We propose a novel approach that leverages the capabilities of the Hungarian algorithm [2] and the Minimum Cost Flow algorithm [3]. By jointly considering the distinct bandwidth needs of users and the limitations of base stations, our solution aims to strike a balance between optimizing bandwidth distribution and ensuring equitable resource allocation. The subsequent sections delve into the technical details of the proposed methodology, highlighting the integration of optimization techniques, and network performance evaluation. The simulation results provide insights into the effectiveness of our approach in comparison to existing resource allocation schemes in MIMO 5G networks. The realm of dynamic bandwidth allocation in 5G MIMO networks has been an active area of research, with numerous studies exploring various strategies for user-to-base station assignment [4], [5], [6], [7], [8].

Our research places a strong emphasis on adaptability, ensuring that the chosen allocation algorithms can respond in real-time to shifting demands, such as fluctuating user populations and evolving application requirements. It differs from the previous ones in the field due to our approach which focuses on adaptability and maintaining peak performance even in the face of fluctuating user populations and evolving application requirements. The Minimum Cost Flow algorithm, in particular, excels in dynamically optimizing bandwidth allocation based on user needs and base station capacities. Furthermore, our approach enhances resource utilization within 5G MIMO networks, intelligently balancing user-to-base station assignments to achieve a more equitable distribution of resources. Through rigorous experimentation and analysis and in order to address the pressing need for efficient, flexible, and future-proof dynamic bandwidth allocation solutions in the complex landscape of 5G MIMO networks, we demonstrate the superiority of our approach in terms of achieving the best allocation strategy, optimizing bandwidth allocation, and advancing the state-of-the-art in dynamic bandwidth allocation [9].

II. Proposed Algorithms

In this section, we introduce a set of algorithms for optimizing user allocation in scenarios with multiple active base stations and multiple different user requirements. The algorithms are tailored to different scenarios and offer varying levels of sophistication. We have already experiment with the techniques in our previous study [10]. In this study as said before we introduce a dynamic scenario that includes a new parameter, the bandwidth requirement from each user. The first algorithm focuses on situations with multiple active base stations. It utilizes a sorting approach, which involves arranging users based on their Signal to Noise Ratio (SNR) value to each base station. The second algorithm is based on the Hungarian algorithm, a powerful combinatorial optimization technique specifically tailored for solving assignment problems. The third algorithm is based on the

Minimum Cost Flow algorithm which provides an effective solution for determining the optimal allocation of users to base stations.

A. Simple Algorithm - User Allocation using only Simple Loops

To further advance our research, we aim to explore user allocation strategies when multiple base stations are active simultaneously. The algorithm's objective is to navigate through an array that stores SNR values for users potentially connecting to multiple active base stations. To achieve this, we will create an array of structures that will include the users. We will iterate through each user in the array and examine if a user with the same user ID already exists in a struct inside the array. If not, the current user will be added to the end of the array. However, if a matching user is found, the code will compare the current user's SNR with the highest SNR value among the matching structures. If the current user's SNR is higher, the corresponding structure in the array will be updated with the current user's information.

Next, we will go through each base station in the array and sort each user's information in descending order based on the SNR value specific to that particular base station. Subsequently, we will create a new structure to store each user's sorted pathloss value, along with corresponding user numbers, base station numbers, distances between base stations and users, and SNR values. Additionally, each base station has a designated capacity, representing the maximum number of users it can accommodate. The algorithm we are employing involves a systematic examination of each base station to determine whether the number of received SNR/user values in a base station surpasses the predefined capacity threshold. When the algorithm encounters pathloss values that do not exceed the capacity threshold, these users are included in the "success" structure. Conversely, if the number of received pathloss values surpasses the capacity limit, the excess values are directed to the "overflow" structure.

In the following pseudocode, a methodical technique to assigning users to base stations in a wireless communication system is described.

Algorithm 1

```
allocatedCapacity_bps = Array of size bs, initialized with zeros
combinedCostMatrix = Array of size numUsers x numBS, initialized with zeros
allocatedStruct = Empty array to store allocated users' information
// Normalize SNR and bandwidth requirements, then calculate combined score
for each user and base station
      for i = 1 to numUsers:
          for currentBaseStation = 1 to bs:
                userIndex = (currentBaseStation - 1) * numUsers + i
                normalizedSNR =
                Normalize(dynamic_pathloss_bs_ue(userIndex).SNR)
                normalizedBandwidth =
                Normalize(dynamic_pathloss_bs_ue(userIndex).Downstream)
                combinedScore = weightSNR * normalizedSNR + weightBandwidth
                * normalizedBandwidth
                combinedCostMatrix[i][currentBaseStation] = combinedScore
// Allocate users to base stations based on combined scores and available
bandwidth
for i = 1 to numUsers:
    bestBaseStation = 0
    sortedBaseStations = SortIndices(combinedCostMatrix[i]) descending
     for j = 1 to numBS:
      currentBaseStation = sortedBaseStations[j]
      userIndex = (currentBaseStation - 1) * numUsers + i
      userRequirementInHZ =  dynamic_pathloss_bs_ue(userIndex).Downstream
      / log2(1 + 10^(dynamic_pathloss_bs_ue(userIndex).SNR / 10))
      if allocatedCapacity_bps[currentBaseStation] + userRequirementInHZ <=
      bscapacity:
      allocatedCapacity_bps[currentBaseStation] += userRequirementInHZ
 // Update allocatedStruct with user information
allocatedStruct.Add({
usernumber: dynamic_pathloss_bs_ue(userIndex).usernumber,
basestation: currentBaseStation,
distance_bs_ue: dynamic_pathloss_bs_ue(userIndex).distance_bs_ue,
pathloss: dynamic_pathloss_bs_ue(userIndex).pathloss,
SNR: dynamic_pathloss_bs_ue(userIndex).SNR,
service: dynamic_pathloss_bs_ue(userIndex).Services,
downstream: dynamic_pathloss_bs_ue(userIndex).Downstream,
upstream: dynamic_pathloss_bs_ue(userIndex).Upstream
})
break // Exit loop after allocation
```

B. Assignment using the Hungarian Algorithm

Our second algorithm serves as an effective solution for addressing linear assignment problems. These problems involve the allocation of rows to columns in a manner that each row is assigned to a column while minimizing the total cost of these assignments. To apply this algorithm to our resource allocation problem, we transform it into a linear assignment problem. Leveraging the Hungarian algorithm, which uniquely matches each row to a column based on their associated costs, we determine the most suitable base station for each user to connect to. By utilizing the M = matchpairs(Cost, CostUnmatched) MATLAB function [11] as Hungarian Algorithm implementation, we solve the linear assignment problem, considering both the rows and columns of the 'Cost' matrix. The 'CostUnmatched' parameter provides the cost associated with not assigning each row to a column and not having a row allocated to each column.

The algorithm normalizes the SNR and bandwidth requirements for each user. Then finds the combined score for each user, using the normalized SNR and the normalized bandwidth multiplying each one with its corresponding weight. The weights are 0.6 for the SNR and 0.4 for the bandwidth meaning that in our algorithms we consider the SNR metric (which is the signal strength of a user to a base station) to have more significance than the requested bandwidth metric, when allocating a user to a base station.

Algorithm 2 – Hungarian Algorithm

```
allocatedCapacity_bps = Array of size bs, initialized with zeros
combinedCostMatrix = Array of size numUsers x (bs * bsAntennas), initialized
with zeros
antennaCounter = 1
minSNR = Minimum SNR value from dynamic_pathloss_bs_ue
maxSNR = Maximum SNR value from dynamic_pathloss_bs_ue
// Normalize SNR & bandwidth, calculate combined score for each user & bs
for i = 1 to numUsers:
   for currentBaseStation = 1 to bs:
      for antennaCounter = 1 to bsAntennas:
        userIndex = (currentBaseStation - 1) * numUsers + i
        normalizedSNR = (dynamic_pathloss_bs_ue(userIndex).SNR - minSNR) /
        (maxSNR - minSNR)
        normalizedBandwidth =
        (dynamic_pathloss_bs_ue(userIndex).Downstream - 1000000) / (25000000
        - 1000000)
        combinedScore = weightSNR * normalizedSNR + weightBandwidth *
        normalizedBandwidth
       columnIdx = (currentBaseStation - 1) * bsAntennas + antennaCounter
       combinedCostMatrix[i][columnIdx] = combinedScore
// Find optimal matching pairs using the Hungarian algorithm
M = HungarianAlgorithm(combinedCostMatrix, 1000)
```

Next, we have to store each base station's user assignments in three initialized data structures. An overflow matrix is built for users who can't be accommodated owing to bandwidth restrictions. According to each iteration of the changed matrix, downstream values are recalculated, and users are assigned to suitable base stations in accordance with their needs. The user's information is added to the appropriate structure if the base station has enough bandwidth; otherwise, the user is flagged for overflow.

Algorithm 2 – Matchpairs Allocation to data structs

```
if basestationw == 1:
  // Allocate users to basestation 1
  bw1 = bw1 - downstream_valueHZ;
  if bw1 > 0:
     assignedtobs1.Add({'usernumber': usernumberw, 'Services':
     dynamic_pathloss_bs_ue(matching_index).Services})
  else:
     overflow.Add([usernumberw, basestationw])
elseif basestationw == 2:
  // Allocate users to basestation 2
  bw2 = bw2 - downstream_valueHZ;
  if bw2 > 0:
     assignedtobs2.Add({'usernumber': usernumberw, 'Services':
     dynamic_pathloss_bs_ue(matching_index).Services})
  else:
     overflow.Add([usernumberw, basestationw])
else:
  // Allocate users to basestation 3
  bw3 = bw3 - downstream_valueHZ;
  if bw3 > 0:
     assignedtobs3.Add({'usernumber': usernumberw, 'Services':
     dynamic_pathloss_bs_ue(matching_index).Services})
  else:
     overflow.Add([usernumberw, basestationw])
second_best_snr_users = Empty Array of structs with fields 'usernumber',
'second_best_snr', 'service', 'bs', and 'reqinHZ'
// Iterate through overflowed users and find second best SNR users
for i = 1 to size(overflow, 1):
     usernumberov = overflow(i, 1);
     current_user_data = Filter dynamic_pathloss_bs_ue where 'usernumber'
     equals  usernumberov
     sorted_data = Sort current_user_data by 'SNR' in descending order
     // Store information of second best SNR user and allocate if possible
      if size(sorted_data) >= 2:
          second_best_snr = sorted_data(2).SNR;
          reqinHZ = Calculate downstream value for second best SNR user
          if bw1 - reqinHZ >= 0 && sorted_data(2).basestation == 1:
             assignedtobs1.Add({'usernumber': usernumberov, 'Services':
             sorted_data(2).Services})
             Set matching row in overflow to zeros
          elseif bw2 - reqinHZ >= 0 && sorted_data(2).basestation == 2:
             assignedtobs2.Add({'usernumber': usernumberov, 'Services':
             sorted_data(2).Services})
              Set matching row in overflow to zeros
          elseif bw3 - reqinHZ >= 0 && sorted_data(2).basestation == 3:
             assignedtobs3.Add({'usernumber': usernumberov, 'Services':
             sorted_data(2).Services})
              Set matching row in overflow to zeros
        elseif size(sorted_data) >= 3:
        end
end
```

C. *Assignment using the Minimum Cost Flow Algorithm*

This study investigates the use of Minimum Cost Flow algorithm to further improve the user allocation and bandwidth distribution process. A Minimum cost flow Algorithm attempts to determine the best possible flow of resources across a network while reducing the overall cost of the flow. In this study, this method can be applied to allocate users to base stations according to their bandwidth demands, signal quality, and network conditions. In this algorithm's code we use the required downstream of the user along with their SNR to find and allocate them to the optimal base station so that the total bandwidth given by all the base stations in the end is maximized. The algorithm also tracks the given bandwidth of each antenna and ensures that no antenna gives more bandwidth than what they are capable of. In cases where a feasible assignment cannot be made due to insufficient bandwidth, the algorithm tracks overflow instances, storing relevant worker information in dedicated matrices. Subsequently, a second iteration is performed for overflowed workers, attempting to find alternative base stations while considering bandwidth constraints. Below is the pseudocode of this algorithm.

Algorithm 2 – Minimum Cost Flow

```
    # Function to find the next available task for a worker
    function find_next_available_task(worker_id, G, assigned_tasks):
       available_tasks = get_available_tasks(G, worker_id) - assigned_tasks
       if available_tasks is not empty:
          sorted_tasks = sort_tasks_by_weight(available_tasks, G, worker_id)
          return sorted_tasks[0]
       else:
          return None
    # Main function to solve the assignment problem
    function solve_assignment_problem(csv_file):
       initialize_bandwidth_variables()
       G = create_graph_from_csv(csv_file)
       max_worker_id = get_max_worker_id(G)
       assigned_tasks = empty_set()
       costs = empty_list()
       task_ids = empty_list()
       pathlosses = empty_list()
       flow_dict = find_min_cost_flow(G)
       overflow, overflow_matrix = process_workers(G, max_worker_id,
assigned_tasks, flow_dict, costs, task_ids, pathlosses)
       print_results(costs, pathlosses, assigned_tasks, overflow,
overflow_matrix)
    # Function to process workers and handle overflow cases
    function process_workers(G, max_worker_id, assigned_tasks, flow_dict,
costs, task_ids, pathlosses):
       overflow = 0
       overflow_matrix = empty_list()
       for worker_id in range(1, max_worker_id + 1):
          task_dict = get_task_dict(flow_dict, worker_id)
          if task_dict is not None:
             task_id = find_next_available_task(worker_id, G, assigned_tasks)
             if task_id is not None:
                assign_task_to_worker(worker_id, task_id, G, assigned_tasks,
costs, pathlosses, task_ids)
             else:
                print(f"No feasible assignment found for Worker {worker_id}")
          else:
             print(f"No feasible assignment found for Worker {worker_id}")
       handle_overflow_cases(G, assigned_tasks, overflow, overflow_matrix)
       return overflow, overflow_matrix
    # Function to assign a task to a worker
    function assign_task_to_worker(worker_id, task_id, G, assigned_tasks,
costs, pathlosses, task_ids):
       edge_data = G[worker_id][task_id]
       task_downstream = edge_data['downstream']
       task_snr = calculate_snr(edge_data['pathloss'])
       downstream_value_HZ = task_downstream / log(1 + pow(10, task_snr /
10), 2)
       if is_bandwidth_available(worker_id, task_id, downstream_value_HZ):
          update_bandwidth(worker_id, task_id, downstream_value_HZ)
          store_assignment_info(worker_id, task_id, G, assigned_tasks, costs,
pathlosses, task_ids)
          remove_assigned_task_edges(worker_id, task_id, G)
       else:
          handle_overflow_case(worker_id, task_id, G, assigned_tasks)
    # Function to handle overflow cases
    function handle_overflow_case(worker_id, task_id, G, assigned_tasks):
       overflow += 1
       print(f"No available bandwidth for Worker {worker_id} and Task
{task_id}")
       overflow_matrix.append({'worker_id': worker_id,
'downstream_valueHZ': downstream_value_HZ})
       remove_assigned_task_edges(worker_id, task_id, G)
    # Function to print the final results
    function print_results(costs, pathlosses, assigned_tasks, overflow,
overflow_matrix):
       num_users = len(costs)
       modified_pathlosses = [your_formula(pathloss, num_users) for pathloss in
pathlosses]
       total_pathloss = sum(modified_pathlosses)
       user_data = total_pathloss / num_users
```

III. Simulation Environment Parameters

Our simulation environment is based on MATLAB and DeepMIMO dataset. The DeepMIMO dataset generation framework has two important features. A Ray-tracing scenario and the parameters for this scenario [12], [13], [14].

A. *DeepMIMO Features*

First, the DeepMIMO channel is created based on precise ray tracing data obtained from Remcom Wireless InSite [14].

Therefore, the DeepMIMO channel captures the dependence on the surrounding geometry/material and transmitter/receiver position. The DeepMIMO dataset generator focused on 5G, generates channel matrices based on the Clustered Delay Line (CDL) channel model defined in 3GPP 38.901 with site-specific statistical distribution parameters obtained from the accurate 3D ray-tracing simulator Remcom Wireless InSite. The software used for the simulation was MATLAB, which generated a channel model that takes into account user path loss and a traffic model that simulates user behavior.

B. Scenario

To evaluate algorithms and accurately simulate real-world scenarios, various parameters must be considered. These parameters include the base station and its capacity, transmit power, noise power, number of transmit antennas (TX) within the base station, users and antenna gain, and the available bandwidth that will be "handed over" to the user. The resources required from different users, are also included in the parameters. So, each user has their own downstream and upstream requirements. The scenario used in this document is the DeepMIMO O1 (Outdoor 1) scenario with an operating frequency of 60 GHz. This scenario consists of 18 base stations and 3 user grids with up to 1,184,923 users distributed across the sitemap (181 users per row) to accurately represent the real-world scenario. Figure 1 shows the scenario sitemap used to create the dataset required for algorithm testing.

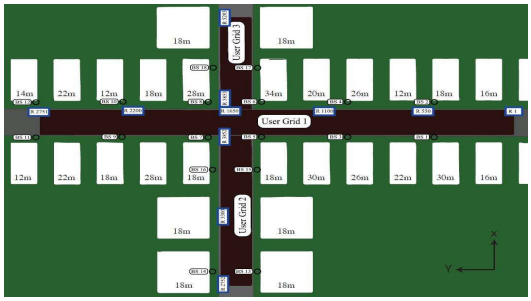

Fig. 1. DeepMIMO O1 scenario outdoors.

Using the parameters shown in Table 1. we calculated the SNR for each user, the average throughput per user and taking into account the downstream/upstream requirements of each user (Table II) along with the bandwidth each base station can give in total, we calculate their optimal allocation to base stations and base station antennas.

TABLE I. SIMULATION PARAMETERS

Simulation Parameters	***Value***
Base Station Bandwidth	400Mhz
Subcarrier Spacing	120 Khz
NRB	264
Active Subcarriers	64
User Gain	0
User Antenna	1
Antenna Gain	21 dBi
Tx Antennas	1000
Transmission Power	45 dBm

TABLE II. USERS BANDWIDTH REQUIREMENTS

Services	***Downstream/Upstream in Mbps***
Browsing/Email	5/2
HDTV	16/0.5
Video Streaming	25/1
Podcasts	2/0.5
VoIP	1/1

The antenna gain is 21 dBi so that the radiation pattern of the antenna may be more efficiently directed or concentrated in a specific direction. The formula for calculating noise power is -174+10 * log10(bandwidth). To increase coverage and serve more consumers, the base station adds 1000 Tx antennas with 45dBm transmission power which is the standard transmission power of a macro cell base station. Each base station has a set user capacity of 1000 and a bandwidth of 400Mhz, corresponding to a real-world scenario in which a single base station supports a heterogeneous group of users with varying signal intensities and requirements. Each user will match with 1 antenna. NRB expresses the transmission bandwidth configuration, in units of resource blocks.

IV. ALGORITHM COMPARISON

In this section, we evaluate several user allocation techniques in MIMO 5G networks, as stated in Section II. First, we assess Algorithm 1, which focuses on allocating users to base stations based on sorting. We present simulated results that show the implications of increasing user connections on throughput, SNR, and network performance. The Hungarian and Minimum Cost Flow algorithms are being implemented to improve user allocation efficiency and to be compared to prior methods. Our goal is to find the best user allocation strategy for MIMO 5G networks.

A. Algorithm 1 – Sorting Algorithm

We run simulations for 181 to 2534 users by selecting the part of the row closest to the active base station. We specifically used the combination from lines 1090 to 1103 for the simulation. Figure 2 and 3 illustrates the number of users assigned to each base station and the bandwidth distribution for each simulation of active users and Table III summarizes the results. The 4th column (overflow) in figure 2 represents the users that didn't connect to a base station because they were all full. It is important to mention that the use of this algorithm has as result some of the users not being able to connect and receive service due to the inefficient management of network resources by the proposed algorithm. As the above table shows in the last two scenarios with 2353 and 2534 users, users with total 126.35Mhz and 202.1Mhz cannot be served by the network due to the non-optimal distribution of the users to the base stations.

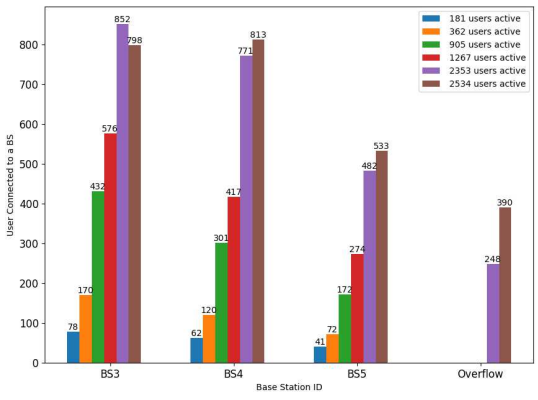

Fig. 2. User Allocation for Algorithm 1

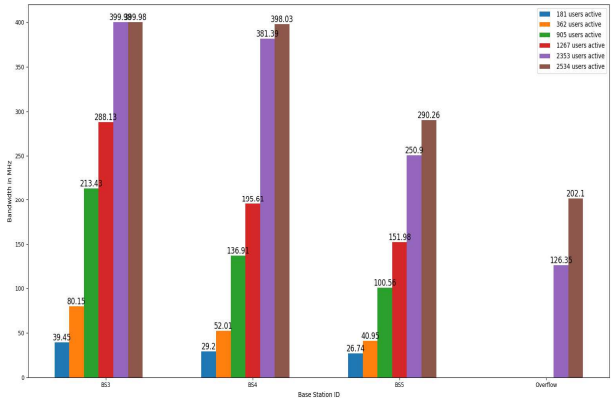

Fig. 3. Bandwidth Distribution for Algorithm 1

TABLE III. ALGORITHM 1 BANDWIDTH

# Users	181	362	905	1267	2353	2534
BW of BS3	39.45 Mhz	80.15 Mhz	213.43 Mhz	288.13 Mhz	399.98 Mhz	399.98 Mhz
BW of BS4	29.2 Mhz	52.01 Mhz	136.91 Mhz	195.61 Mhz	381.39 Mhz	398.03 Mhz
BW of BS5	26.74 Mhz	40.95 Mhz	100.56 Mhz	151.98 Mhz	250.9 Mhz	290.26 Mhz
Average BS BW	31.78 Mhz	57.7 Mhz	150.3 Mhz	211.9 Mhz	344.09 Mhz	362.76 Mhz
Overflow BW	0 Mhz	0 Mhz	0 Mhz	0 Mhz	126.35 Mhz	202.1 Mhz

B. *Algorithm 2 – Hungarian Algorithm*

As with our previous simulations, we run experiments for 181, 362, 905, 1267, 2353 and 2534 users. This function enabled experimental analysis and proved to be a major advance in research. The Hungarian algorithm function showed remarkable efficiency in solving the assignment problem of assigning users to base stations. Through careful optimization, this method achieved significant improvements in average throughput per user and bandwidth allocation as well as a better distribution of users through the base stations. The results are particularly noticeable in Figure 4 and 5, showing a more balanced and fair distribution of users and bandwidth across base stations. Table V presents the results. Using this algorithm, we have unassigned users only in the 2534 users scenario and the unassigned users are only 94 comparing with the previous algorithm which has 390 unassigned users. Moreover, using this algorithm we bandwidth of the unassigned users is only 94Mhz compared to 202.1Mhz of the previous algorithm.

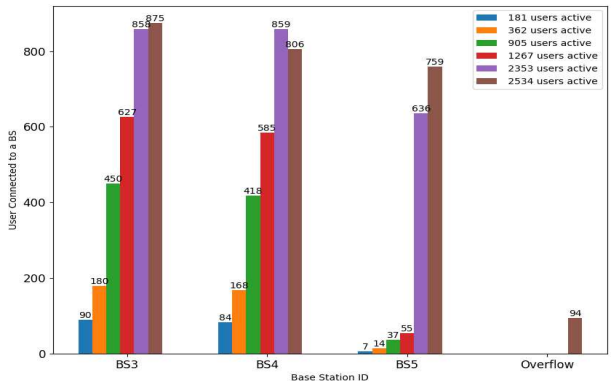

Fig. 4. User Distribution for Hungarian Algorithm

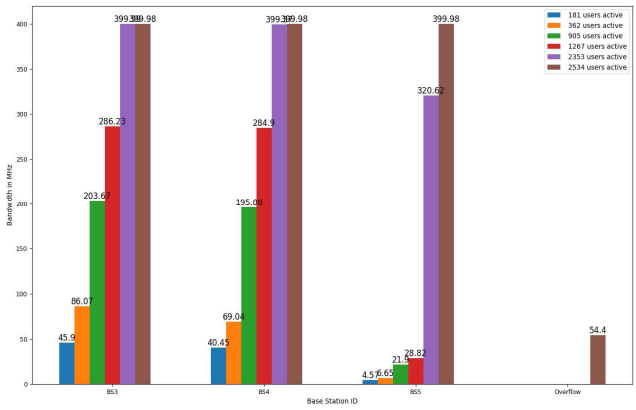

Fig. 5. Bandwidth Distribution for Hungarian Algorithm

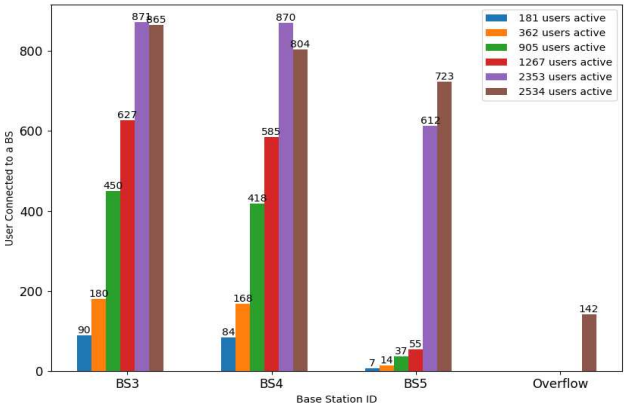

Fig. 6. User Distribution for Minimum Cost Flow Algorithm

TABLE IV. HUNGARIAN ALGORITHM BANDWIDTH

# Users	181	362	905	1267	2353	2534
BW of BS3	45.9 Mhz	86.07 Mhz	203.67 Mhz	286.23 Mhz	399.98 Mhz	399.98 Mhz
BW of BS4	40.45 Mhz	69.04 Mhz	195.87 Mhz	284.89 Mhz	399.98 Mhz	399.98 Mhz
BW of BS5	4.56 Mhz	6.65 Mhz	21.9 Mhz	28.82 Mhz	320.62 Mhz	399.98 Mhz
Average BS BW	30.30 Mhz	53.92 Mhz	140.48 Mhz	199.98 Mhz	373.53 Mhz	399.98 Mhz
Overflow BW	0 Mhz	0 Mhz	0 Mhz	0 Mhz	0 Mhz	54.4 Mhz

C. *Algorithm 3 – Minimum Cost Flow Algorithm*

Running our simulations with 181, 362, 905, 1267, 2353 and 2534 users the Minimum Cost Flow algorithm showed great efficiency in maximizing the per-user throughput as well as achieving a user distribution close to the one achieved with the Hungarian algorithm. Figures 6, 7 show the user allocation results and the bandwidth distribution for Minimum Cost Flow. Table V summarizes the bandwidth results. Using this algorithm, we have unassigned users only in the 2534 users scenario and the unassigned users are 142. Comparing with the previous algorithms, this algorithm is better than the first algorithms but performs worse than the second algorithm. Moreover, using this algorithm we bandwidth of the unassigned users is only 75,97Mhz. Again, comparing with the previous algorithms, this algorithm is better than the first algorithms but performs worse than the second algorithm.

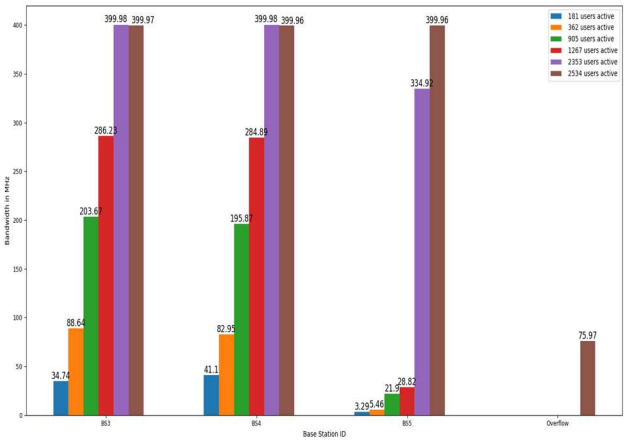

Fig. 7. Bandwidth Distribution for Minimum Cost Flow Algorithm

TABLE V. MINIMUM-COST FLOW BANDWIDTH

# Users	181	362	905	1267	2353	2534
BW of BS3	34.74 Mhz	88.64 Mhz	203.67 Mhz	286.23 Mhz	399.99 Mhz	399.97 Mhz
BW of BS4	41.1 Mhz	82.95 Mhz	195.87 Mhz	284.89 Mhz	399.98 Mhz	399.96 Mhz
BW of BS5	3.29 Mhz	5.46 Mhz	21.9 Mhz	28.82 Mhz	334.92 Mhz	399.96 Mhz
Average BS BW	26.38 Mhz	59.02 Mhz	140.48 Mhz	199.98 Mhz	378.3 Mhz	399.96 Mhz
Overflow BW	0 Mhz	0 Mhz	0 Mhz	0 Mhz	0 Mhz	75.97 Mhz

The overall comparison between the three algorithms shows us that in cases of 181, 362, 905, 1267 users, all three algorithms exhibited a similar performance. However, as the user count increased to 2353 and 2534, notable distinctions emerged. The Minimum Cost Flow and Hungarian algorithms demonstrated superior per-user throughput compared to the simple allocation method. Additionally, the Hungarian algorithm exhibited the least number of overflowed users, emphasizing its capability to achieve a more even distribution among base stations. This aligns with the algorithm's intrinsic preference for equitable resource allocation. In contrast, the Minimum Cost Flow algorithm, exhibited slightly more overflowed users, as it prioritizes maximizing throughput for each user and the total given bandwidth. Comparing the total given bandwidth across the three algorithms, the Minimum Cost Flow algorithm allocated the most, followed by the Hungarian algorithm and then the simple allocation.

V. CONCLUSION & FUTURE WORK

In conclusion, the evaluation of resource allocation algorithms, including the simple allocation, Hungarian, and Minimum Cost Flow algorithms, has provided valuable insights into their respective strengths and limitations. For scenarios involving a large user base, all algorithms exhibited comparable results in terms of the percentage of users attaining their requested bandwidth. However, as the user count decreased, the Hungarian, and Minimum Cost Flow algorithms demonstrated superior per-user throughput, with the Hungarian algorithm notably excelling in achieving a more even distribution among base stations and minimizing overflowed users. The Minimum Cost Flow algorithm exhibited a slightly higher incidence of overflowed users due to its emphasis on maximizing individual user throughput and total given bandwidth. Overall, the findings underscore the importance of aligning the choice of resource allocation algorithm with the specific objectives and requirements of the given scenario, as each algorithm's design priorities significantly influence its performance outcomes. These insights can be practically leveraged by system designers to tailor their choice of resource allocation algorithm. Concluding, incorporating machine learning algorithms to address overflow issues presents a promising avenue for future research and practical implementation, offering the potential to further enhance the adaptability of resource allocation systems in real-world settings.

ACKNOWLEDGMENT

This research has been co-financed by the Hellenic Foundation for Research & Innovation (H.F.R.I) through the H.F.R.I.'s Research Projects to Support Faculty Members & Researchers (project code: 02440).

REFERENCES

[1] N. H. M. Adnan, I. M. Rafiqul and A. H. M. Z. Alam, "Massive MIMO for Fifth Generation (5G): Opportunities and Challenges," 2016 International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, 2016, pp. 47-52.

[2] Kuhn, Harold W. "The Hungarian method for the assignment problem." Naval research logistics quarterly 2.1 - 2 (1955): 83-97.

[3] Goldberg, Andrew V. "An efficient implementation of a scaling minimum-cost flow algorithm." Journ. of algorithms 22.1 (1997): 1-29.

[4] M. Manini, C. Gueguen, R. Legouable and X. Lagrange, "Study of MIMO Channel Matrices Correlation to Optimize Resource Allocation Algorithms in Multi-Users 5G," 2019 12th IFIP Wireless and Mobile Networking Conference (WMNC), Paris, France, 2019, pp. 162-166, doi: 10.23919/WMNC.2019.8881567

[5] K. Keshav, A. K. Pradhan, T. Srinivas and P. Venkataram, "Bandwidth allocation for interactive multimedia in 5G Networks," 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India, 2021, pp. 840- 845, doi: 10.1109/ICCES51350.2021.9488978.

[6] S. Yoshioka, S. Suyama, T. Okuyama, J. Mashino and Y. Okumura, "Digital beamforming algorithm for 5G low-SHF-band massive MIMO with intersite coordination", 2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC), Bali, Indonesia, 2017, pp. 470-475, doi: 10.1109/WPMC.2017.8301859

[7] S. Suyama, J. Mashino, Y. Kishiyama and Y. Okumura, "5G multi-antenna technology and experimental trials," 2016 IEEE International Conference on Communication Systems (ICCS), Shenzhen, China, 2016, pp. 1-6, doi: 10.1109/ICCS.2016.7833602

[8] M. A. I. Sarder, F. Tajrian, M. Rafique, M. Anzum and A. B. Shams, "Configuring Antenna System to Enhance the Downlink Performance of High Velocity Users in 5G MU-MIMO Networks", 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 2021, pp. 1-6, doi: 10.1109/ACMI53878.2021.9528218.

[9] K. N. R. S. V. Prasad, E. Hossain and V. K. Bhargava, "Energy Efficiency in Massive MIMO-Based 5G Networks: Opportunities and Challenges," in IEEE Wireless Communications, vol. 24, no. 3, pp. 86-94, June 2017, doi: 10.1109/MWC.2016.1500374WC.

[10] C. Bouras, D. Diasakos, A. Gkamas, V. Kokkinos, P. Pouyioutas and N. Prodromos, "Evaluation of User Allocation Techniques in Massive MIMO 5G Networks," 2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM), Istanbul, Turkiye, 2023, pp. 1-6, doi: 10.1109/WINCOM59760.2023.10322955.

[11] Matchpairs function documentation in Matlab: [Online].Available: https://www.mathworks.com/help/matlab/ref/matchpairs.html

[12] DeepMIMO Dataset. [Online]. Available: http://www.DeepMIMO.net

[13] A. Taha, M. Alrabeiah and A. Alkhateeb, "Deep Learning for Large Intelligent Surfaces in Millimeter Wave and Massive MIMO Systems," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013256

[14] Remcom Wireless Insite. [Online]. Available: https://www.remcom.com/wireless-insite-em-propagation-software