RnD Project
A Game Theoretic Approach to Optimal Network Allocation

Prajval Nakrani : 17D070014
Parth Shettiwar : 170070021
Rathod Harekrissna : 17D070001
Guide: Prof. Prasanna Chaporkar

March 2020

Contents

1 Motivation

In the modern world, using cellular communication has become a necessity and with growing demand for network consumption, the number of base-stations providing the facility has also increased considerably. With this huge number of users and base-stations providing the service, choosing an optimal allocation of all users to base stations has no longer remained a simple problem. There are various research work done along similar problems already like one approach used by Prof. Prasanna Chaporkar in his paper where is solves the joint problem of Cell Zooming and Channel Allocation.

Utilizing this opportunity, we decided to try and solve this problem as a Research and Development project under Prof. Prasanna Chaporkar. We start with defining the problem statement, then we analyse the complexity of the problem - the problem turns out to be NP-Hard. In order to achieve globally optimum solution, we can not do much better than brute-force if we want a deterministic algorithm to find the solution. We had initially tried to solve it using a greedy approach, but we could easily come up with a small sized counter example where in the greedy approach produces a result that is a local optimum but not a global optimum. Hence, motivated from approach used by Prof. Prasanna Chaporkar in his paper on Joint Cell Zooming and Channel Allocation, we also decided to formulate our problem as a Potential Game and solve it using a probabilistic approach, that is Spatial Adaptive Play (SAP). This approach adds a certain degree of randomness to regular BRD algorithm so that it doesn't get stuck in local optima.

2 Problem Statement

There are N number of Base-Stations (BS) and P number of users. The jth BS allocates C_j resources per packet. If the user i connects to BS j, he gets Rij number of packets. User i can connect to a limited set of BS given by the set Ui.

All the entries of Cj, Rij are non-negative. You know the vector $[C_j]$ $(1 <= j <= N)$, 2D-matrix $[R_{ij}]$ $(1 <= j <= N, 1 <= i <= P)$, set of sets $[U_i]$.

Any user can be connected to at max a BS. An allocation vector x is defined as a vector where $x[i]$ denotes the BS user i will connect to. Also it should be ensured all users are connected to a BS. You need to find an allocation such that, the sum of average resource allocated by a base station to its users over all BS is maximized (Optimal Allocation). More formally, you need to find an allocation that maximizes

$$F(x) = \sum_{j=1}^{N} \sum_{i=1}^{P} \frac{C_j R_{ij} I_{ij}}{N_j}$$

Note that $I_{ij} = 1$ if User i is connected to BS j, I_{ij}=0 otherwise and N_j is the number of users connected to BS j. Also note that the 2-D matrix I_{ij} can be easily constructed from the given allocation vector x by initializing I matrix with zeros and simply looping over elements of x and keeping $I[i][x[i]] = 1$. Hence knowing an allocation vector x is equivalent to knowing the 2-D matrix I, which is also equivalent to knowing the set of sets $[U_i]$. The following section contains an example describing the problem statement

2.1 An example describing problem statement

Say N=4, P=2; Also we know resources allocated per base station as:

$$C = \begin{pmatrix} 10 & 10 & 10 & 2 \end{pmatrix}$$

Implying $C_1 = 10, C_2 = 10, C_3 = 10, C_4 = 2$;
The R matrix given is:

$$R = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

U_1 = [1, 2, 3] – means U1 can connect to BS 1, 2, or 3; U_2 = [3, 4] – means U2 can connect to BS 3, or 4
Optimal allocation is : x = [1, 3] – U_1 connects to BS 1 and U_2 connects to BS 3.
The value of required maximized sum = 30

3 First Approach

We first tried using a greedy approach to solve the problem. Here, we optimize $F(x)$ constructively until we reach a global optimum. The greedy used algorithm is as described below :

3.1 Greedy Algorithm

1. Produce a matrix X of size $P \times N$, where, $X_{ij} = R_{ij}C_jI_{ij}$
2. Flatten the matrix X into a list Y and sort the list Y in descending order, all while maintaining the index of each element it had in the X matrix. This index would later serve to assign base-stations to users.
3. Assign the first BS-user pair in the list as per the index. And calculate $F_{current} = F(x)$
4. $F_{old} = F_{current}$
5. Move on to the next element, re-calculate $F_{current} = F(x)$, assuming this BS-user pair in allocation.
6. If$(F_{current} > F_{old})$: Accept the allocation. Else: $F_{current} = F_{old}$
7. Go to step 4

However, we could easily come up with a counter example that shows that this problem doesn't converge to the global optimum. The counter example is follows:

N = 2, P = 2. Both users can connect to both the base-stations.

$$C = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} 200 & 198 \\ 100 & 3 \end{pmatrix}$$

Here, the optimal allocation vector is $A = (2, 1)$ (meaning user 1 is connected to base station 2 and user 2 is connected to base station 1). However, our algorithm produces the result, $A = (1, 2)$. So, we can see that the greedy approach fails terribly and will produce a local optimum in most of the cases. So, next we move on to solving the problem using a Game Theory approach, where we formulate the problem as a potential game.

4 Game Formulation

First we define the users as the players of a game. That means there are P players in the game. Each player can and must thus be connected to a Base-station. BS are numbered from 1 to N, changing U_i accordingly. Define the set of players as A. Noting that there are N BS and each user can be connected to one of the BS in it's connectivity set U_i, define the strategy of a player as the number of BS it is connected to. Hence a player can choose his strategy from the set U_i. Note that this is a finite game as each of the players can choose a strategy from a finite set U_i.

We define the strategy of player i with symbol x_i and hence $x_i \in U_i$. Also, we define the set containing the strategy of all the players as x. Hence, $x = [x_1, x_2, ... x_P]$. We will also call x as a state from now on. Also $x \in U$, where U is the state space or set of all possible strategies of all players. Also note that given a state x, we can compute all the values of the 2D matrix $\{I_{ij}\}$ and also all the values of 1D matrix N_j. Hence, state information is equivalent to knowing $\{\{I_{ij}\}, \{N_j\}\}$. Hence $x \equiv \{\{I_{ij}\}, \{N_j\}\}$. Now define the set x_{-j} as the set of strategies of all players except the player j. Hence, $x = [x_j, x_{-j}]$. We also define the set of all the possible strategies x_{-j} as X_{-j}

Now note that we want to maximize the function:

$$\phi(x) = \sum_{j=1}^{N} \sum_{i=1}^{P} \frac{C_j R_{ij} I_{ij}}{N_j} \tag{1}$$

and hence we want to set up the utility function of the players in such a way so that the resulting game is an exact potential game with the function to be maximized as the exact potential function. Let the utility function of all the players be u_i. For defining the utility function of all the players, first we need to define neighbours and neighbour set of a user.

4.1 Defintions

4.1.1 Definition 1: Neighbours of a User

A user j is said to be a neighbour of user i if and only if $U_i \cap U_j \neq \phi$. Note that i maybe equal to j. This also implies that there is at least a BS which both users i and j can potentially connect to.

4.1.2 Definition 2: Neighbour set of a User i ($\mathcal{N}_i$):

Set of all the neighbour of a user i.
More formally, $\mathcal{N}_i = \{j : \exists\ k \in U_j$ such that $k \in U_i\}$ or $\mathcal{N}_i = \{j : U_i \cap U_j \neq \phi\}$

4.1.3 Definition 3: Welfare function of a User i:

We define the **welfare function** of each user as follows:

$$f_i(x) = \sum_{j=1}^{N} \frac{C_j R_{ij} I_{ij}}{N_j} \tag{2}$$

with terms having usual meaning

4.1.4 Definition 4: Utility function of a User i:

We define the **utility function** of each user u_i as follows:

$$u_i(x) = \sum_{z \in \mathcal{N}_i} \sum_{j=1}^{N} \frac{C_j R_{zj} I_{zj}}{N_j} \tag{3}$$

with terms having the usual definition.

Note: Both welfare function and utility function are functions of strategy state x.

4.1.5 Definition 5: Local Interaction

A set of finite valued functions (interpreted as welfare functions) $\{f_j(\cdot)\forall j\}$ is said to satisfy the local interaction property if, $\exists \mathcal{N}_k \subseteq \mathcal{N}$ such that, $\forall i \notin \mathcal{N}_k$ and $x_k, x'_k \in U_k$,

$$f_i(x) = f_i\left(x_k, x_{-k}\right) = f_i\left(x'_k, x_{-k}\right)$$

This means action of player k would not affect welfare of any non-neighbouring player i. Note: Here $\mathcal{N}$ are the set of all users.

4.1.6 Definition 6: Selection scheme (C,p) for C-SAP

A selection scheme consists of $(C,\mathbf{p})$, where set $C = C_1, C_2, ..., C_M$ denotes a collection of sets $C_k \subset N$ such that,

1. For all distinct $i, j \in \mathcal{C}_k$, we have $i \notin \mathcal{EN}_j$, $\mathcal{EN}_j$ denotes the set of two hop neighbours of j. We call such a set $\mathcal{C}_k$ as a two-hop independent set.

2. $\forall j \in \mathcal{N}$, there exists a k such that, $j \in \mathcal{C}_k$. That is, every node is present in at least one of the classes.

3. $\mathbf{p}_k > 0$ denotes the probability of picking the set $\mathcal{C}_k$

4.2 Propositions

4.2.1 Proposition 1:

The neighbours of a node defined by the above definition satisfy the local interaction property as defined above for the defined welfare function for the nodes.

Proof: The term I_{ij} takes care of whether which node is connected to which Base Station. Now, we know that a particular user cannot connect to all the base stations. This is given by the U_i set. Therefore, we know that $I_{ij} = 0$ $\forall\ j \notin U_i$. Therefore, effectively, the welfare function boils down to $f_i(x) = \sum_{j \in U_i} \frac{C_j R_{ij} I_{ij}}{N_j}$. The only way a player k can affect the welfare function of another user i is by affecting the parameter N_j in the function $f_i(x)$, since all other parameters of that user are either already fixed (C_j, R_{ij}) or depends on his action only (I_{ij}). Now we know that for non neighbouring nodes $U_i \cap U_k = \phi$. Hence action of a player non-neighbouring player k cannot affect the parameter N_j $\forall$ $j \notin U_i$ also since it has no base station common with the user i. Thus, there is no relation between the welfare functions of two non-neighbouring nodes. And hence, a change is one doesn't affect the change in other, as is expected from the local interaction property.

4.2.2 Proposition 2:

$\phi(x_j, x_{-j}) - \phi(x_j^{'}, x_{-j}) = u_j(x_j, x_{-j})$ - $u_j(x_j^{'}, x_{-j})$, $\forall j \in \mathcal{N}$, $\forall x_j, x_j^{'} \in U_j$, $\forall x_{-j} \in X_{-j}$

Proof : The $\phi(x)$ function is nothing but the sum of welfare functions of all the users. From the Local Interaction property, we know that welfare functions of non-neighbouring users do not affect each other. Now, the utility function for a particular user is the sum of welfare functions of all the nodes belonging to its Neighbour set. Now, if the user changes its action, the welfare functions of only its neighbours will be affected and which are already included in the utility function. Thus, if a player changes its action, the difference in the $\phi(x)$ is only due to the difference in the utility function of that particular user as non-neighbour nodes are unaffected.

4.3 Algorithms

4.3.1 Best Response Dynamics

Given : $\mathcal{N}$ (Set of all the users), $u_i(x)$ $\forall i \in \mathcal{N}$
Aim : To obtain a Nash Equilibrium of the potential game

1. Initialise the strategy set $x = [x_1, x_2, ...x_{\mathcal{N}}]$ with $x_j \in U_j$ $\forall j \in \mathcal{N}$
2. Select a random user i from set of all users $\mathcal{N}$ uniformly
3. Compute $u_i(x, X_{-i})$ $\forall x \in U_i$
4. Set x_i as the strategy that maximizes the u_i from the above computed values
5. Repeat from step 2.

4.3.2 SAP Algorithm

Given : $\mathcal{N}$ (Set of all the users), $u_i(x)$ $\forall i \in \mathcal{N}$, β(Learning Rate)
Aim: To achieve the strategy set x or the allocation vector of base stations to every user such that ϕ is maximised.

1. Initialise the strategy set $x = [x_1, x_2, ...x_{\mathcal{N}}]$ with $x_j \in U_j$ $\forall j \in \mathcal{N}$
2. Select a random user i from set of all users $\mathcal{N}$ uniformly
3. Compute P$(x_i) = e^{\beta u_i(x_i, X_{-i})}$ $\forall$ $x_i \in U_i$
4. Normalise P(x_i) to a valid pmf where $x_i \in U_i$
5. Sample a strategy s from pmf P(x_i).
6. Update the strategy set x, by replacing user i previous strategy x_i by sampled strategy s.
7. Go to step 2.

4.3.3 C-SAP Algorithm

Given: $\mathcal{N}$, $u_i(x)$ $\forall i \in \mathcal{N}$, C, β(Learning Rate)
Aim: To achieve the strategy set x or the allocation vector of base stations to every user such that ϕ is maximised.

1. Randmly sample a cluster C_k from the set of clusters C.
2. For all players in C_k, update all their strategies, as given in SAP Algorithm from steps 3-6, simultaneously.

4.3.4 Algorithm for generating C_k

Given: $\mathcal{N}$, $\mathcal{N}_j$
Aim: To make C_k a two-hop independent set.

1. $\mathcal{C}_k \leftarrow \emptyset, \mathcal{B} \leftarrow [1, 1, 1, ...P$ times]
2. sum = sum of elements of $\mathcal{B}$
3. if sum = 0 : return current C_k; break;
4. prob = $\mathcal{B}/sum$
5. j = random index sampled from probability distribution prob

6. $C_k \leftarrow C_k \cup j$
7. $\forall i \in \mathcal{E}N_j$ make $\mathcal{B}[i] = 0$
8. jump to step 2

4.4 Theorems

4.4.1 Theorem 1

The game satisfying local interaction property where an action of a player can only affect welfare of its neighbours and the utility(payoff) of a player is just the summation of welfare function over its neighbours is an exact potential game with summation of welfare functions over all users as the potential function.

Proof: For a game to be exact potential game, the changes in payoffs of a user should track the changes in potential function exactly. This was exactly shown in Proposition 1 for function $\phi(x)$. Hence we show that the game is a Potential game with $\phi(x)$ as the Potential Function for the problem.

4.4.2 Theorem 2

For every β >0, the Spatial Adaptive Process defines a Markov Chain on the State Space U with a unique stationary distribution given by Gibbs Distribution:

$$\pi_\beta(x) = \frac{e^{\beta\phi(x)}}{\sum_{z\in U} e^{\beta\phi(z)}} \tag{4}$$

Proof: We will divide the proof into 2 parts: 1)Existence and 2)Uniqueness:

1. *Existence of Stationary Distribution:*
We will show a stronger result, the proposed $\pi_\beta(x)$ satisfies the detailed balance state condition, i.e. it satisfies:

$$\pi_\beta(x)P_{xy} = \pi_\beta(y)P_{yx}\forall x, y \in U \tag{5}$$

For simplicity we will write $\pi_\beta(x)$ as $\pi(x)$ from now onwards.
First we observe that $P_{xy} = P_{yx} = 0$ unless $y = x$ or y and x differ at exactly one position. This is because at any iteration, we can update the strategy of a single user, which can lead him to take either the same previous strategy (leading to same overall strategy set) or can change to different strategy pertaining to set U_i of that user i (leading to differ in overall strategy set at one position). Say they differ at position j, hence $x_j \neq y_j$ and $x_{-j} = y_{-j}$. Also the selection of a user at each step is uniform with probability $\frac{1}{P}$ where P is the total number of users. Hence we can find P_{xy} as :

$$P_{xy} = \frac{1}{P}\frac{e^{\beta u_j(y_j,x_{-j})}}{\sum_{x_z\in U_i} e^{\beta u_j(x_z,x_{-j})}}$$

Now writing $\pi_\beta(x)P_{xy}$:

$$Ke^{\beta u_j(y_j,x_{-j})+\beta\phi(x)}$$

where K:

$$\frac{1}{P\sum_{x_z\in U_i} e^{\beta u_j(x_z,x_{-j})}\sum_{z\in U} e^{\beta\phi(z)}}$$

Now:

$$Ke^{\beta u_j(y_j,x_{-j})+\beta\phi(x)} = Ke^{\beta u_j(y_j,x_{-j})-\beta u_j(x_j,x_{-j})+\beta u_j(x_j,x_{-j})+\beta\phi(x)} = Ke^{\beta\phi(y_j,x_{-j})-\beta\phi(x_j,x_{-j})+\beta u_j(x_j,x_{-j})+\beta\phi(x)}$$

$$= Ke^{\beta\phi(y_j,x_{-j})+\beta u_j(x_j,x_{-j})} = \pi(y)P_{yx}$$

In above we used result from Proposition 2
This clearly implies it satisfies the stationary distribution since:

$$\sum_{x\in U}\pi(x)P_{xy} = \sum_{x\in U}\pi(y)P_{yx} = \pi(y)\sum_{x\in U}P_{yx} = \pi(y) \tag{6}$$

2. *Uniqueness of stationary Distribution:*
   Since the Markov chain constructed is irreducible, the stationary distribution has to be unique. This is because we know that $\mu_i\pi_i = 1$ where μ_i is the mean time to return to state i. This implies whenever a stationary distribution exits we can find the μ_i mean return time as:
$$\mu_i = \frac{1}{\pi_i} \tag{7}$$
   Now since mean return times for every state have to be unique, hence π_i will also be unique implying the above defined stationary distribution as gibbs distribution is unique for the formulated Markov Chain.

   Proof for (7):
   We can write $\mu_i\pi_i$ as:
$$\mu_i\pi_i = E(T_i|X_0 = i)P(X_0 = i)$$
   where T_i is the first time to return to state i. X_0 is the initial state. Now we can rewrite this as:
$$\mu_i\pi_i = \sum_{n=1}^{\infty} P(T_i \geq n|X_0 = i)P(X_0 = i) = \sum_{n=1}^{\infty} P(T_i \geq n, X_0 = i)$$
   For n=1, we get $P(T_i \geq n, X_0 = i) = P(X_0 = i)$. Also for n$\geq$ 2, we get:
$$P(T_i \geq n, X_0 = i) = P(X_{n-1} \neq i, X_{n-2} \neq i, ..., X_1 \neq i, X_0 = i)$$
   which can be modified a little to finally get $\mu_i\pi_i$ as:
$$\mu_i\pi_i = P(X_0 = i) + \sum_{n=2}^{\infty}(P(X_{n-1} \neq i, X_{n-2} \neq i, ..., X_1 \neq i) - P(X_{n-1} \neq i, X_{n-2} \neq i, ..., X_1 \neq i, X_0 \neq i))$$
$$= P(X_0 = i) + \sum_{n=2}^{\infty}(P(X_{n-2} \neq i, X_{n-3} \neq i, ..., X_0 \neq i) - P(X_{n-1} \neq i, X_{n-2} \neq i, ..., X_1 \neq i, X_0 \neq i))$$
   We shifted the index since the process is stationary.
   Now this forms a telescopic sum to finally give :
$$\mu_i\pi_i = P(X_0 = i) - \lim_{n\to\infty} P(X_{n-1} \neq i, X_{n-2} \neq i, ..., X_0 \neq i) + P(X_0 \neq i)$$
   The limit term goes to 0 as $n \to \infty$, since we know for sure that all states are recurrent, i.e. they can be reached with a non-zero probability always or $P(V = \infty|X_0 = i) = 1$ where V is the total number of visits to state i. The above term wants to find the probabilty of not returning to state i, which is 0. Hence we get:
$$\mu_i\pi_i = 1 \tag{8}$$

4.4.3 Theorem 3

For a fixed selection scheme (C,p), the C-SAP algorithm defines a Markov Chain on the state space S, with a unique stationary distribution given by the Gibbs Distribution

Proof : We prove this theorem in a manner similar to the proof of Theorem 2, we will prove that π satisfies the detailed balance equation. The only difference is that of the expression of P_{xy} in C-SAP algorithm.
But before that we note that the welfare function of a user depends on its state and the utility function of the user depends on the state of its neighbours. Therefore we can say that
$$u_j(\mathbf{x}) = u_j(\mathbf{x}_k, \mathbf{x}_{-k}) = u_j(\mathbf{x}'_k, \mathbf{x}_{-k}), \text{ if } k \notin \mathcal{EN}_j \tag{9}$$
Next, we need to show that,
$$\pi(\mathrm{x})\mathrm{P}_{\mathrm{xy}} = \pi(\mathrm{y})\mathrm{P}_{\mathrm{yx}}$$
We observe that,

1. The chain is irreducible because every node gets to update with a non-zero probability
2. The chain is aperiodic because self-transition can happen with non-zero probability

With the above observations, we can say that the markov chain has a unique stationary distribution π, which is the solution of the detailed balance equation.
We introduce a couple of notations below

- I_{xy} denotes the set of indices where state x and y differ.
- $A_{xy} = k : I_{xy} \subseteq C_k$

Hence, the transition $x \to y$ can happen only via the selections present in the set A_{xy}. Also note that, $A_{xy} = A_{yx}$ When $x \to y$ is not feasible, then $A_{xy} = \Phi$ and $P_{xy} = P_{yx} = 0$ and the equation holds trivially.
Now we move on to the case when $x \to y$ is feasible. Then,

$$P_{xy} = \sum_{k \in \mathcal{A}_{xy}} p(k) \Pr\{x \to y | \mathcal{C}_k \text{ was selected }\} \tag{10}$$

where,

$$\Pr\{x \to y | \mathcal{C}_k\} = \prod_{j \in \mathcal{C}_k} \Pr\{j \text{ chooses action } y_j | x\} \tag{11}$$

where,

$$\Pr\{j \text{ chooses } y_j | \mathrm{x}\} = \frac{e^{\beta u_j(y_j, \mathrm{x}_{-j})}}{\sum_{z \in \mathcal{S}_j} e^{u_j(z, x_{-j})}} = \frac{e^{\beta u_j(y_j, \mathbf{x}_{-j})}}{\sum_{z \in \mathcal{S}_j} e^{u_j(z, y_{-j})}}$$

This above equality holds true due to (9). Multiplying (10) by $\pi(x)$ and substituting in above equation, we get

$$\boldsymbol{\pi}(x) \Pr\{x \to y | \mathcal{C}_k\} = \frac{1}{D_1 D_2} e^{\beta\left\{\phi(x) + \sum_{j \in \mathcal{C}_k} u_j(y_j, y_{-j})\right\}}$$

$$= \frac{1}{D_1 D_2} e^{\beta\left\{\sum_{j \in \mathcal{C}_k} u_j(x_j, y_{-j}) + \phi(y)\right\}}$$

$$= \pi(y) \Pr\{y \to x | \mathcal{C}_k\} \tag{12}$$

where D1 is the denominator from (4) and D2 is the product of denominators from (11). The transition from the first to second expression is achieved by exchanging the welfare functions from $\phi(x)$ with respective utility functions from the set C_k. Hence, by using equation (12), (10) and the fact that $A_{xy} = A_{yx}$, we prove that the detailed balance equation holds true.

5 Implementation

We approached to do the simulations of the above algorithms by ensuring randomization in the input. First we select Number of Base Stations(N) and Number of users(P) randomly. Similarly we randomly assign resources to the Base Stations and get the entries of C vector. Lastly For finding the R and I connection matrix we use the following scheme to closely simulate the real world scenario:

5.1 Scheme for simulating Rate and Connection Matrix

1. First we make a 10000×10000 area plot, where we place our Base Stations and Users.
2. To simulate the real world case, we place our decided number of base stations(N), uniformly across the plot built.
3. Now the decided number of users are placed randomly over the plot.
4. Then compute the distances of each user from every base station and decide whether the user can connect to that base station using the following expression:

   $$I_{ij} = [\frac{K}{D_{ij}^2} > Threshold]$$

   where D_{ij} is the distance between the user i and base station j. K and Threshold are the hyperparamters, I is the connection matrix as defined in the problem statement
5. We now get the entries for Rate matrix R as :

   $$R_{ij} = \begin{Bmatrix} \frac{K}{D_{ij}^2}, & \text{if } I_{ij} = 1 \\ 0, & \text{else} \end{Bmatrix}$$

Having simulated the input, we apply on it, Three Algorithms, which were discussed in Section 4.3, namely:

1. Best Response Dyamics
2. Spatial Adaptive Play(SAP)
3. Concurrent Spatial Adaptive Play

Now to perform step 5 in SAP Algorithm, i.e. to sample a strategy from pmf P(x_i), we used the Inverse Transform Sampling Method as our sampling strategy:

5.2 Inverse Transform Sampling

Given a discrete random variable X such that we know $P(X = x_i) = p_i$ for every x_i, we can sample a strategy using the following approach:

1. Generate $U{\sim}Unif(0,1)$.
2. Determine the index k such that $\sum_{j=1}^{k-1} p_j{\le}U{\le}\sum_{j=1}^{k} p_j$.
3. Return $X = x_k$

We use this selection scheme where ever we want to sample from a given probability distribution which is not uniform, like in Scheme for making C_k clusters, in SAP, CSAP algorithm for selecting an allocation from set of possible allocation after finding their probability distribution and so on.

5.3 Scheme for making $\mathcal{E}N$ set

For making the required $\mathcal{E}N$ set, we use the popular Breadth First Search algorithm (BFS), where in we find the nearest and next nearest neighbours of all nodes given a graph (Connections). The exact algorithm is as described below:

```
1. initialize $\mathcal{E}N$ as a set of sets with $P$ number of empty set.
2. loop for all users (x) staring from 1$^{st}$:
   • initialize array h with $P$ number of 0
   • initialize a queue Q with starting user (say x)
   • while Q is non empty, loop:
     (a) cur = Q.top
     (b) if h[cur] = 3: break;
     (c) for all nodes (j) loop:
         i. if (connections[cur][j] = 1) and (h[j] = 0): h[j] = h[j] + 1; Q.append(j)
   • for all nodes (i) loop:
     (a) if (0<h[i]<=3): $\mathcal{EN}[x]$.append(i)
```

After completing this process, you will get the complete $\mathcal{EN}$ set, which can be used for making C_k using the scheme described above. After getting the C_k set, we are ready to use the CSAP algorithm along with the others

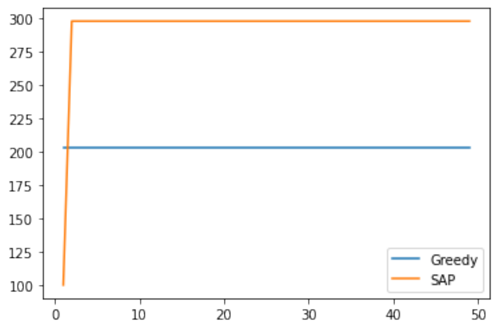
5.4 Hyper-Parameter tuning

For computing the probabilties in SAP Algorithm, as in Step 4, we use a learning rate parameter β. Instead of keeping it constant, we vary it as $c \times \sqrt{iter - number}$, where c is a constant and $iter - number$ is the number of iterations happened till that time. This ensures a higher exploration at initially since β will be small, hence all strategies will have chance to be selected. At higher iteration number, the beta increases and the sampling algorithm favours the strategy which produces maximum utility. Another possibility is to vary β as the natural logarithm of the number of iterations but our simulation size being less, we cannot use the log function effectively as the number of iterations involved will be too less and SAP algorithm will never move on to more deterministic stage.

Having implemented all the functions, we move on to perform the simulations to get an approximate picture of the real time working of the three algorithms.

6 Simulation Results

First we simulate our handcrafted example as mentioned in section 3, for which our greedy (BRD) algorithm gets stuck at local optimum. We simulate both regular BRD and SAP algorithm and see how the results differ.



N=2, P=2, C,R as in section 3

As we observe in the above figure that regular BRD gets stuck at local optima but SAP algorithm only settles at the global maximum. Hence proving the validity of the devised frameworks of the algorithms

We perform various simulations on the three algorithms viz. BRD, SAP and C-SAP to cover all the cases for values of Number of Base stations (N), Number of People (P) and Threshold. Specifically we plot the graph of potential function(ϕ) vs Number of Iterations and look at the convergence rate and convergence values for the Three Algorithms. For defined values of the parameters (N,P,Threshold) forms a triplet that completely defines the simulation environment. We define Low and High value for each of the parameters as defined in the table below.

	Low	High
N	16	81
P	400	1000
Threshold	10	100

Table 1: Parameters Values

We consider 8 combinations of parameters with Low and High values as shown in the table above for simulation which give 8 (N,P,T) triplets and compare the performance of all the three algorithms.

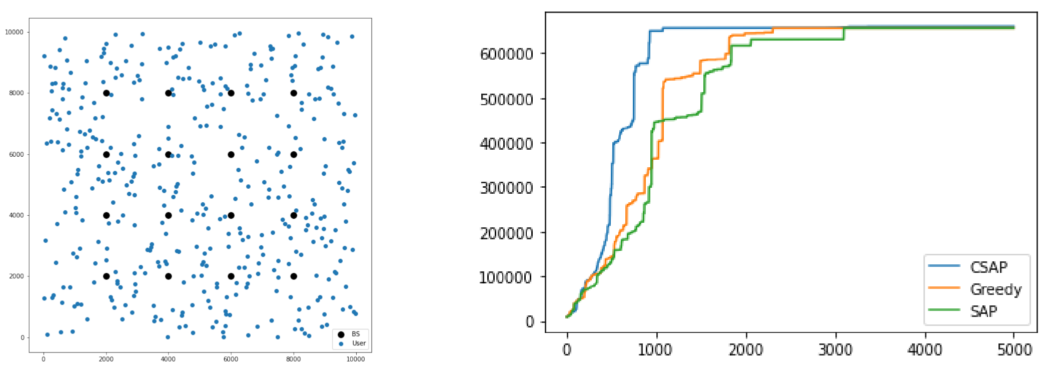
A lower Threshold would imply users can easily connect to far-off base stations and vice versa. Different values of N and P are taken to simulate the real world scenarios of Crowded and Uncrowded places.
We set the values for hyper-paramters K, Defined in Section 6.1 and c, Defined in Section 6.2, as 10000 and 10 respectively. Further we perform all simulations for either 5000 or 10000 iterations, depending on when the algorithms converge.

For each of the above cases we also show the Plot of position of Base Stations and Users, where Users are completely placed randomly on the 10000×10000 plot and Base Stations are placed uniformly. Finally we also plot a case for N=1024 and P=1000, a hypothetical case to simulate the scenario when Number of Base stations are almost equal to the Number of Users present.

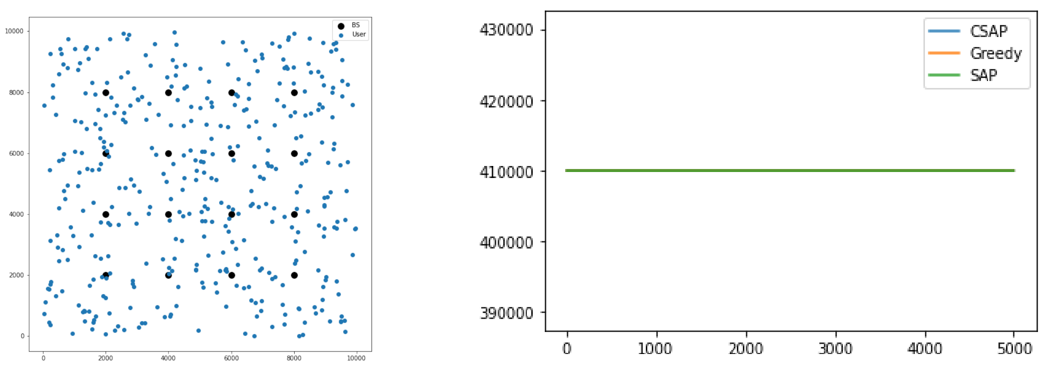
Sr. No.	N	P	Threshold	Avg. Cluster Size
1	16	400	10	1.42
2	16	1000	10	1.36
3	81	400	10	1.12
4	81	1000	10	1.249
5	16	400	100	16
6	81	400	100	12.04
7	81	1000	100	12.56
8	1024	1000	100	1.23

Table 2: C-SAP cluster sizes for all (N,P,T) combinations

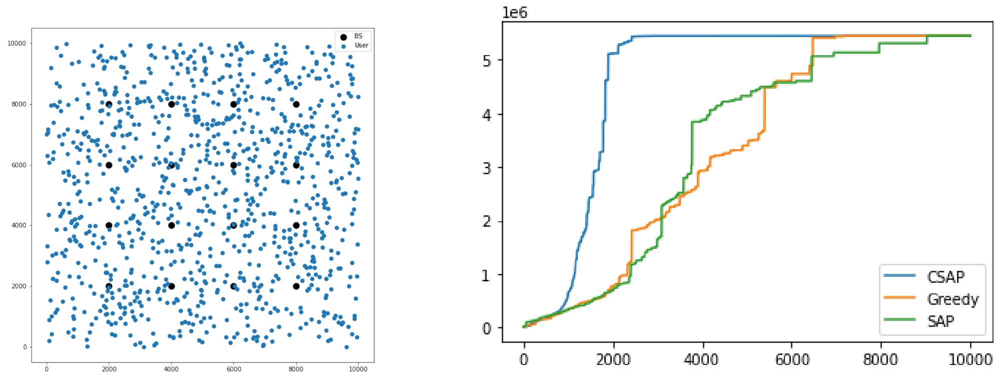
Note: In all the following graphs, BRD has been labeled as Greedy



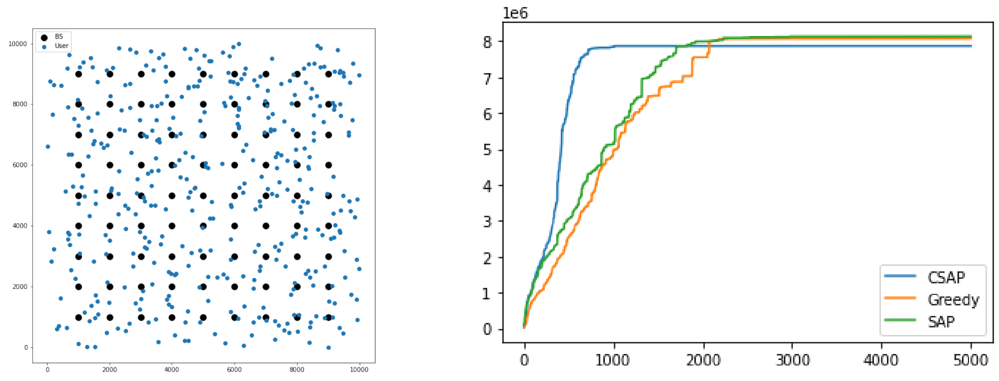
(a)



(b)

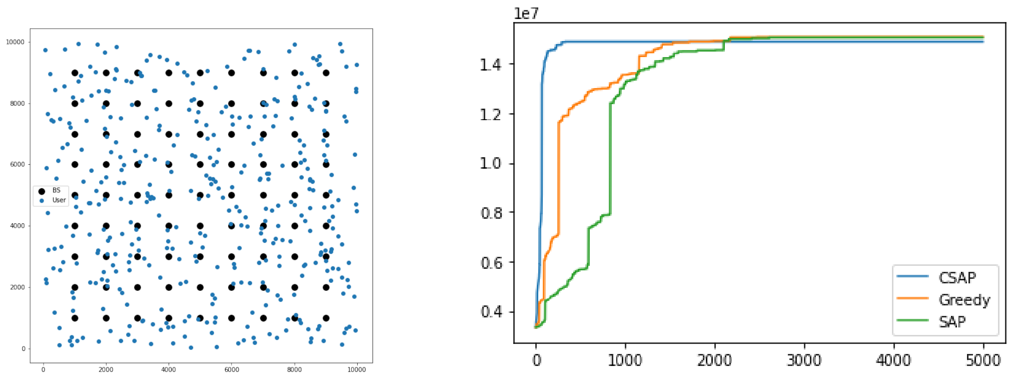


(c)

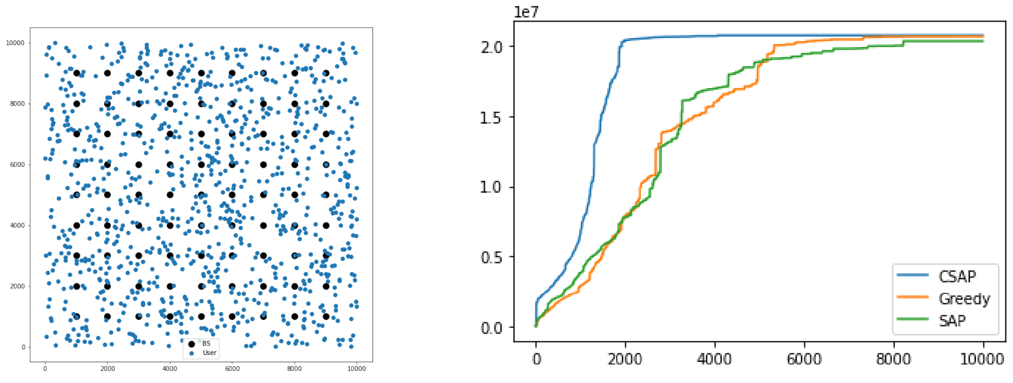


(d)

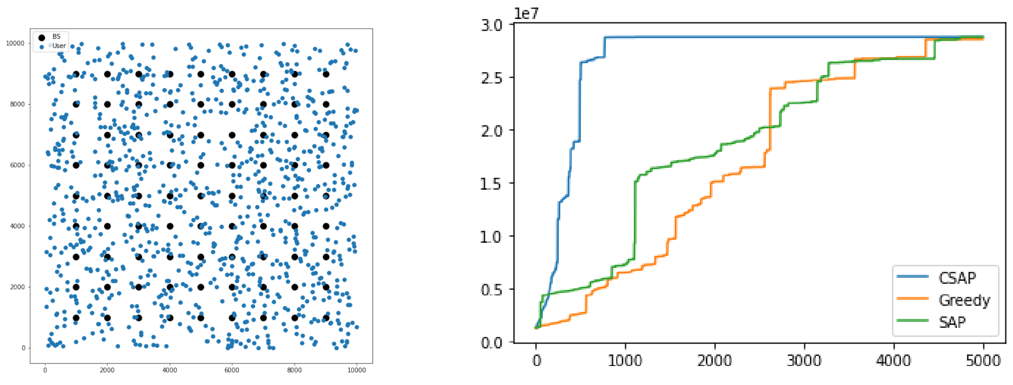
Figure 1: (a)N=16, P=400, Threshold=10 (b)N=16, P=400, Threshold=100 (c)N=16, P=1000, Threshold=10 (d)N=81, P=400, Threshold=10. The plots on Left are Base Stations and Users coordinates in the area and the plots on right are the Potential function vs iterations plotted for three algorithms



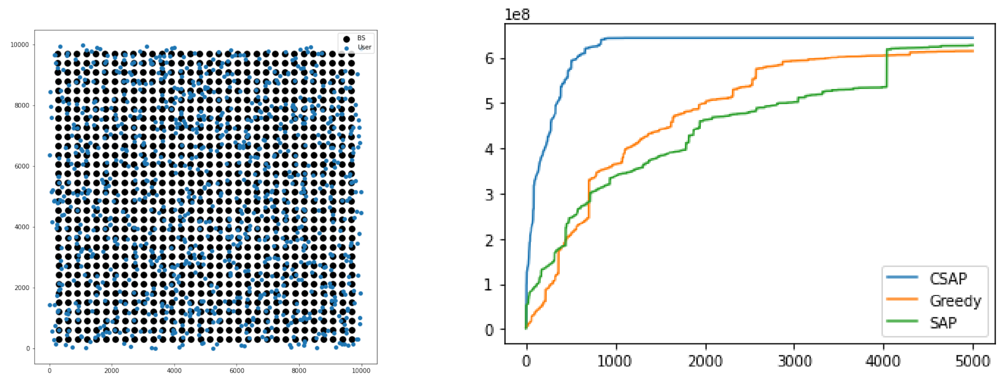
(a)



(b)



(c)



(d)

Figure 2: (a)N=81, P=400, Threshold=100 (b)N=81, P=1000, Threshold=10(c)N=81, P=1000, Threshold=100 (d)N=1024, P=1000, Threshold=100. The plots on Left are Base Stations and Users coordinates in the area and the plots on right are the Potential function vs iterations plotted for three algorithms

7 Observations from Simulations

1. As we see, C-SAP converges faster than regular BRD or SAP algorithm in all scenarios. Main reason for this is simultaneous update of all users within a cluster.
2. For second case with (N,P,T) = (16,1400,100), the number of base stations are few and hence are placed sparsely and a threshold of 100 causes the radius of reach of base station to become very small, hence there is no overlap between the coverage of any 2 base stations, as a result, each user has only one option to connect to the base station nearest to it. Hence, the solution is already obtained in that case and hence we see a flat graph for potential function value v/s iterations.
3. In most of the cases, in case of all the 3 algorithms, the potential function value converges to almost the same optima. This observation would be ”normal” if it were only SAP and C-SAP, but BRD also almost always converges to same value as SAP. This means that there are not many inefficient Nash Equilibiria for the potential function for our set-up.
4. In the last hypothetical senrario case of N=1024 and P=1000, observe again the same case where CSAP converges faster than the other two algorithms.

8 Conclusion

In this work, we have tried to approach a NP Hard real world Base Station Allocation problem. The whole problem is formulated using a Game Theory Framework . We explore existing three Algorithms in the literature and prove their convergence using various theorems and definitions. We implement them using various schemes with a set of hyper-parameters list which control them. The Simulation of Algorithms is done using a randomized input which mimics the real world case. A comparison of three algorithms is done on various possible configurations of Number of Base Stations, Number of Users and The powers of Base stations and plot their overall potential function. We finally conlcude that irrespective of configuration, CSAP beats the other two algorithms in terms of amount of time to converge and converge to a global optima given sufficient time.

9 Future Work

The problem described in work is a smaller prototype of the problem in real world which is much more complicated, where we have channels allocation issues for neighbouring base stations and each Base Station faces many hardware constraints. In the current problem statement, we have assumed a fixed and equal coverage for every base-station which is decided by the threshold that is manually selected by the user. However, in reality, Every Base station can operate at different transmit powers. As a result, we can incorporate two Real-Life complications into our problem statement to make the model more realistic. They are as follows:

1. Each base station has can vary its transmit power and hence change its coverage independent of other base stations. Now, in the final allocation, every base station needs to select a transmit power such that there is no user that is left unserved and such that the allocation is optimum.
2. Next, each base station transmits data over various channels. In reality, if two base station with physical coverage overlap transmit data over the same channel, the users will face data collision. Hence, the allocation should be such that two base stations whose physical coverage overlaps, should not transmit data over a common channel. They should transmit data over mutually exclusive channel sets.

Including these two variables in the allocation problem greatly increases the complexity of problem. The formulation of the game will change in this case, however, it can still be solved using a similar approach.