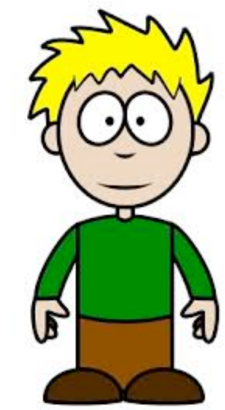
Approximating the FDMA Capacity

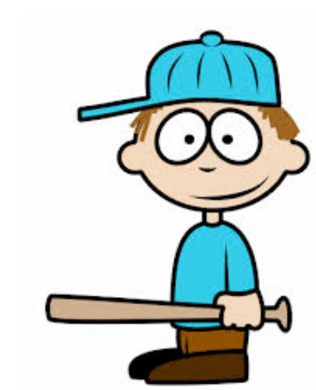
Rahul Vaze

Kiran Koshy

Andrew Thangaraj

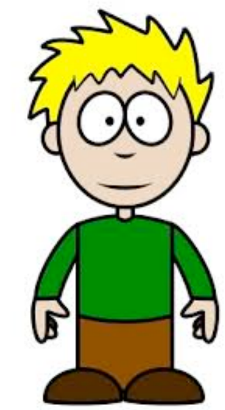
Partitioning Problem

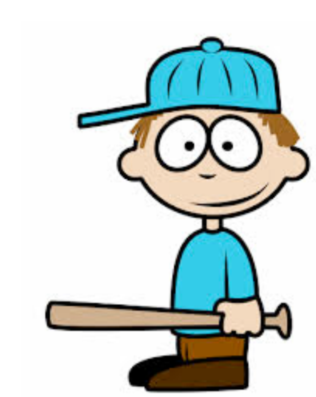




Partitioning Problem

S_i



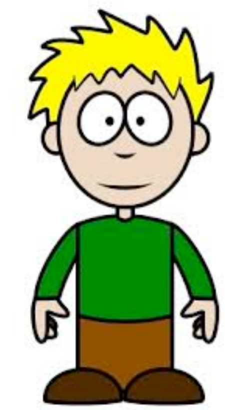


Partitioning Problem

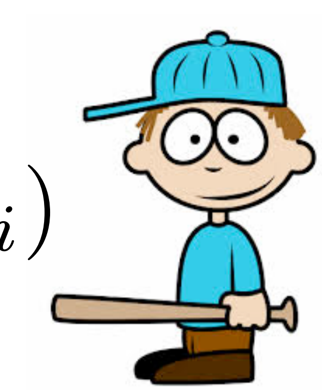
$f_1(S_i)$

S_i

$f_2(S_i)$



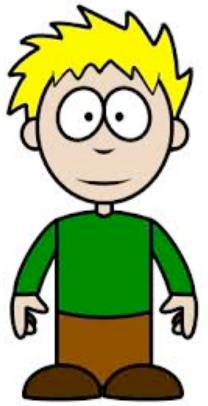
$f_n(S_i)$



Partitioning Problem

$f_1(S_i)$

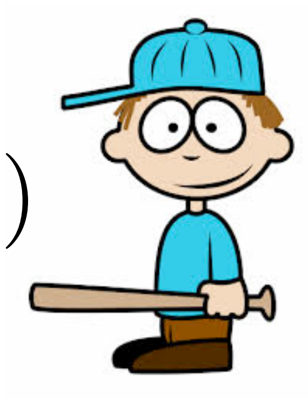
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$

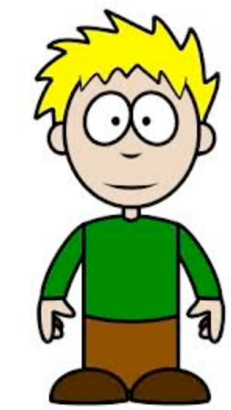


Partitioning Problem

$f_1(S_i)$

$$max \sum_{i=1}^{n} f_i(S_i)$$

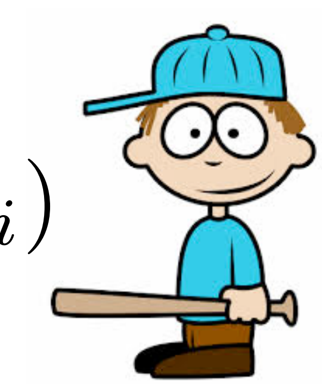
$f_2(S_i)$



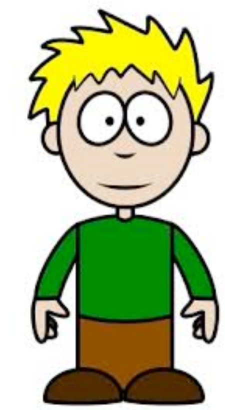
Partition S

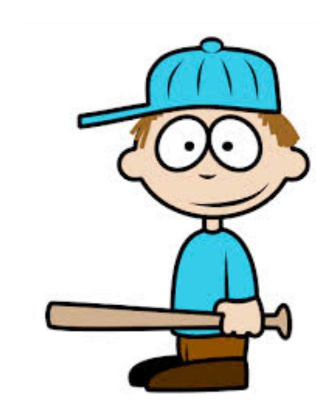
into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



Greedy Algorithm - Partitioning Problem

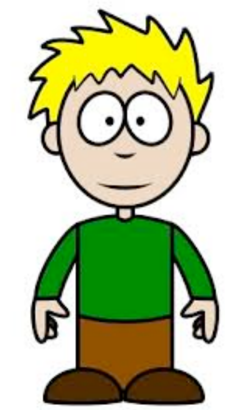




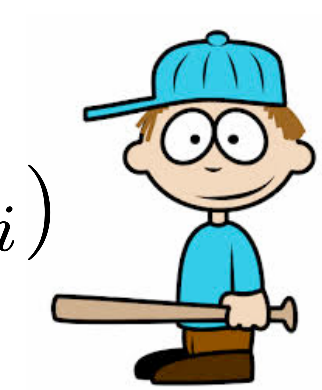
Greedy Algorithm - Partitioning Problem

$f_1(S_i)$

$f_2(S_i)$



$f_n(S_i)$



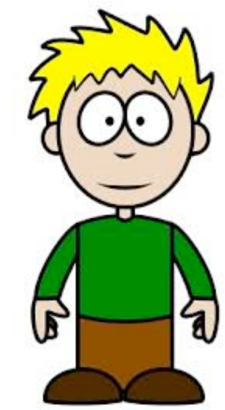
Greedy Algorithm - Partitioning Problem

Partition S

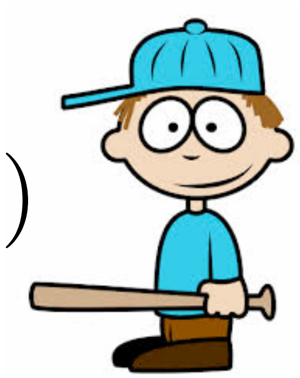
into $S_1, S_2, \ldots, S_n$

$f_1(S_i)$

$f_2(S_i)$



$f_n(S_i)$

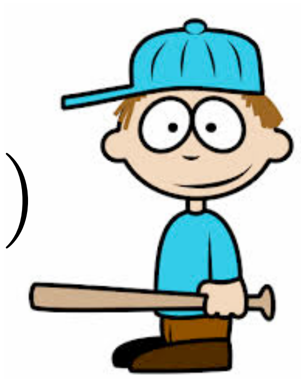


Greedy Algorithm - Partitioning Problem

s_1

$f_1(S_i)$

$f_2(S_i)$



$f_n(S_i)$

Partition S

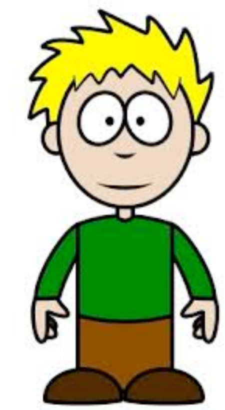
into $S_1, S_2, \ldots, S_n$

Greedy Algorithm - Partitioning Problem

$f_1(S_i)$

s_1

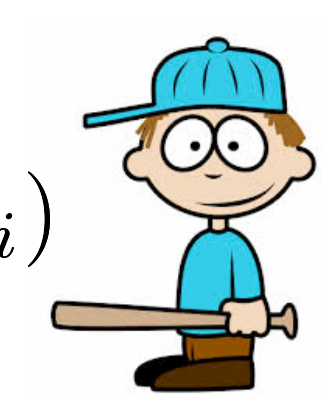
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$

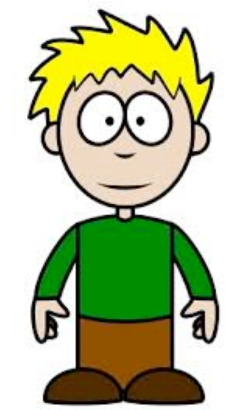


Greedy Algorithm - Partitioning Problem

$f_1(S_i)$

s_1

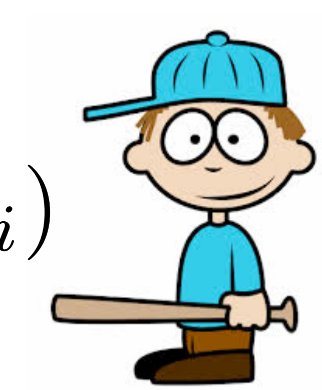
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



Greedy Algorithm - Partitioning Problem

$f_1(S_i)$

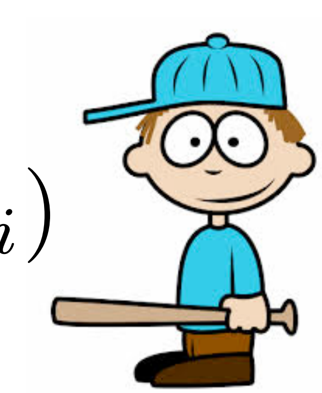
s_1

$f_2(S_i)$

Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



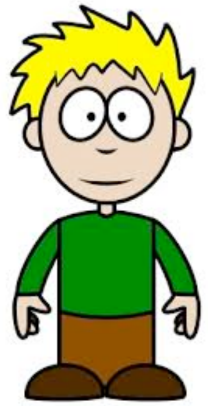
Greedy Algorithm - Partitioning Problem

s_1

$f_1(S_i)$

s_1

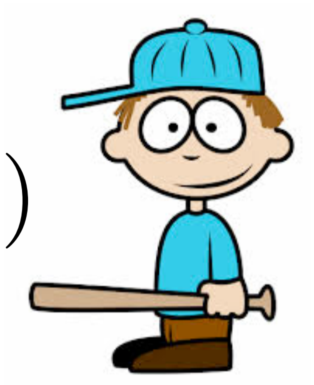
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



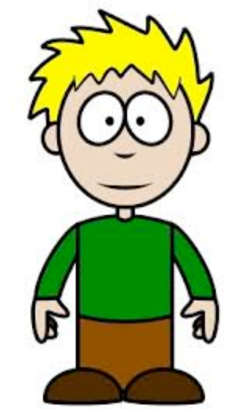
Greedy Algorithm - Partitioning Problem

s_1

$f_1(S_i)$

s_2

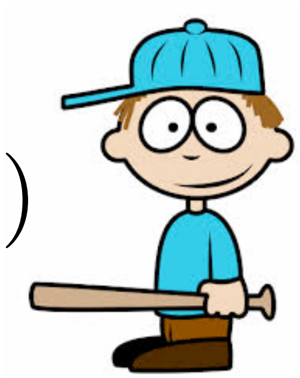
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



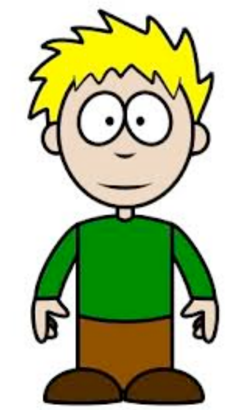
Greedy Algorithm - Partitioning Problem

s_1

$f_1(S_i)$

s_2

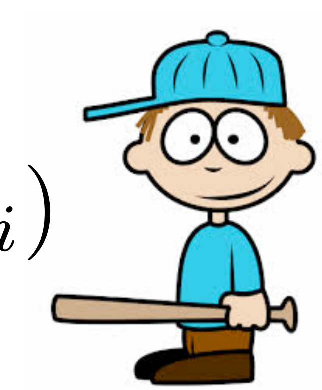
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



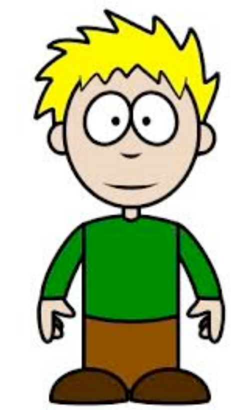
Greedy Algorithm - Partitioning Problem

s_1

$f_1(S_i)$

s_2

$f_2(S_i)$

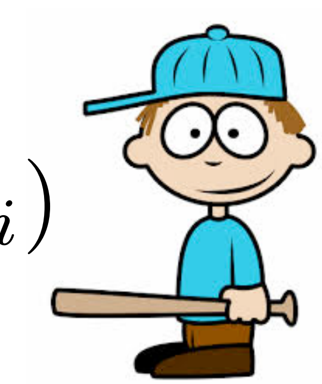


Partition S

into $S_1, S_2, \ldots, S_n$

s_2

$f_n(S_i)$



Guarantee

Greedy Algorithm : *At each step add an element*

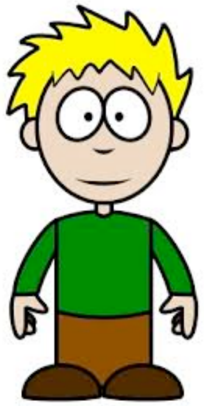
that maximizes the incremental gain.

Guarantee

Greedy Algorithm : *At each step add an element that maximizes the incremental gain.*

Theorem (Nemhauser et. al. 1978): *If each f_i is monotone and sub-modular, then the greedy algorithm achieves at least 1/2 fraction of the optimal solution.*

Greedy Algorithm is actually online

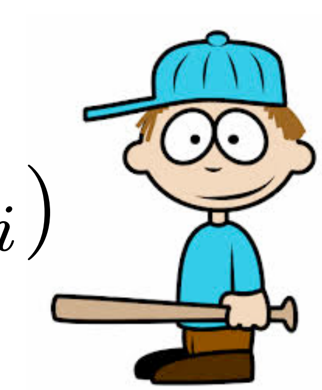


Greedy Algorithm is actually online

$f_1(S_i)$

$f_2(S_i)$

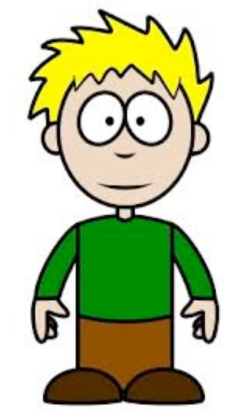
$f_n(S_i)$



Greedy Algorithm is actually online

$f_1(S_i)$

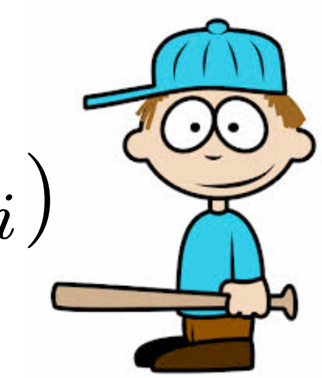
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$

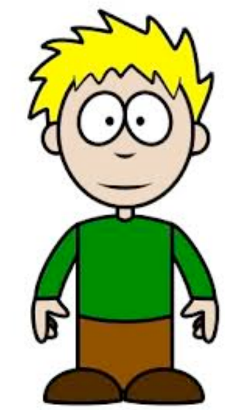


Greedy Algorithm is actually online

$f_1(S_i)$

s_1

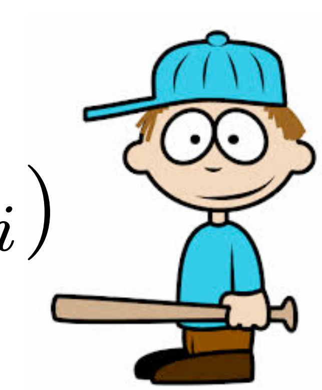
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$

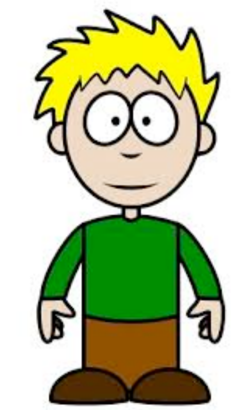


Greedy Algorithm is actually online

$f_1(S_i)$

s_1

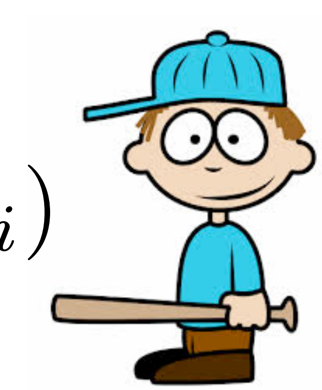
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$

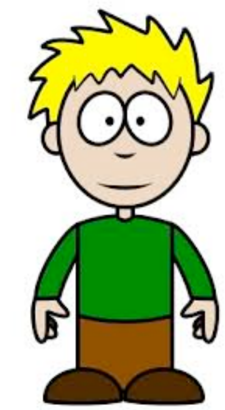


Greedy Algorithm is actually online

$f_1(S_i)$

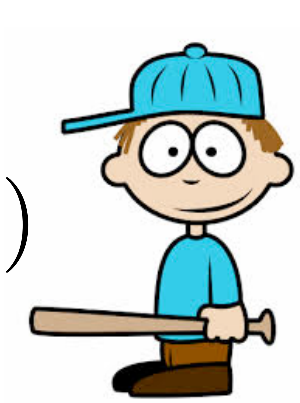
s_1

$f_2(S_i)$



Partition S
into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$

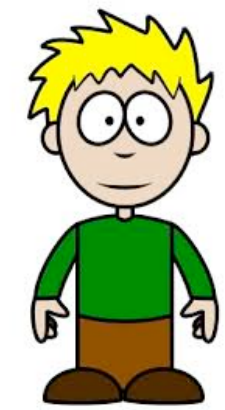


Greedy Algorithm is actually online

$f_1(S_i)$

s_1

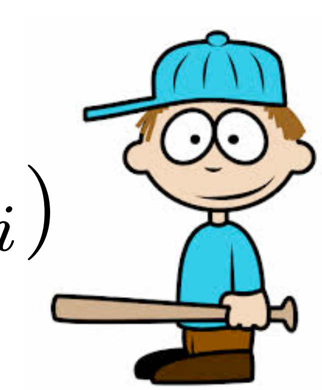
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



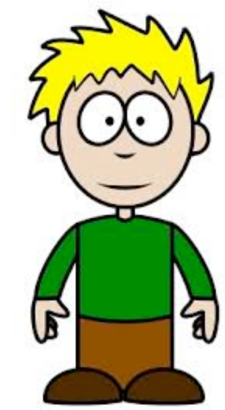
Greedy Algorithm is actually online

s_1

$f_1(S_i)$

s_1

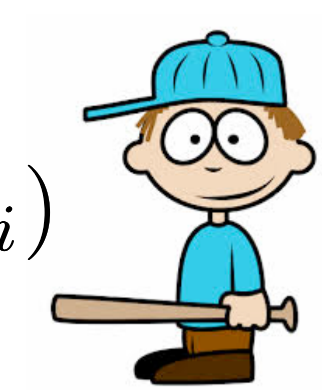
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



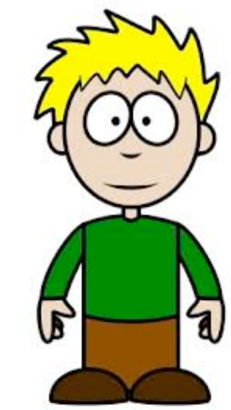
Greedy Algorithm is actually online

s_1

$f_1(S_i)$

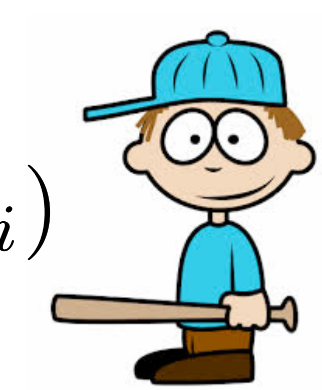
s_2

$f_2(S_i)$



Partition S
into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



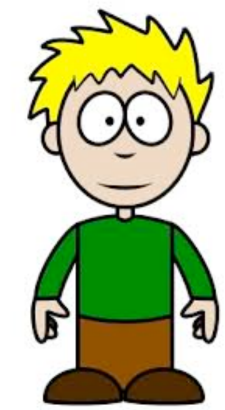
Greedy Algorithm is actually online

s_1

$f_1(S_i)$

s_2

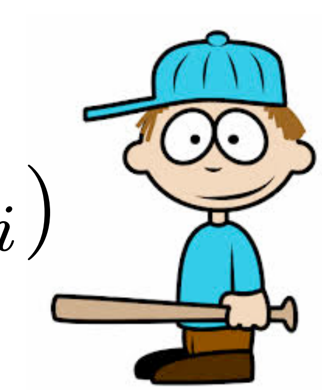
$f_2(S_i)$



Partition S

into $S_1, S_2, \ldots, S_n$

$f_n(S_i)$



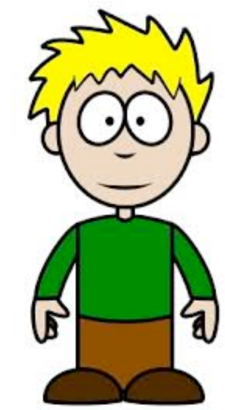
Greedy Algorithm is actually online

s_1

$f_1(S_i)$

s_2

$f_2(S_i)$

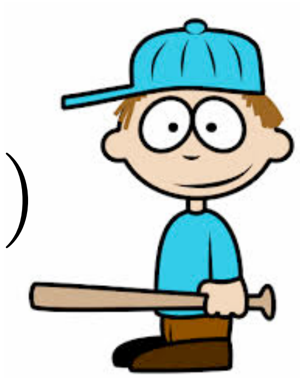


Partition S

into $S_1, S_2, \ldots, S_n$

s_2

$f_n(S_i)$



Guarantee

Online Greedy Algorithm : *At each step add an element that maximizes the incremental gain.*

Guarantee

Online Greedy Algorithm : *At each step add an element that maximizes the incremental gain.*

Theorem *(Nemhauser et. al. 1978): If each f_i is monotone and sub-modular, then the online greedy algorithm achieves at least 1/2 fraction of the optimal solution.*

Some Priliminaries

Some Priliminaries

$\mathrm{L}et\ f : 2^U \rightarrow \mathbb{R}$

Some Priliminaries

Let $f : 2^U \to \mathbb{R}$

Then f is called *monotone* if $f(S \cup \{a\}) \geq f(S)$. $S \subset U$, $a \in U$

Some Priliminaries

Let $f : 2^U \to \mathbb{R}$

Then f is called *monotone if* $f(S \cup \{a\}) \geq f(S)$. $S \subset U, \ a \in U$

Then f is called *sub-modular if*

Some Priliminaries

$\mathrm{L}et\ f : 2^U \to \mathbb{R}$

Then f is called *monotone if* $f(S \cup \{a\}) \geq f(S).\ \ S \subset U,\ \ a \in U$

Then f is called *sub-modular if*

$$f(S \cup \{a\}) - f(S) \geq f(T \cup \{a\}) - f(T),\ \ S \subseteq T.$$

Some Priliminaries

Let $f : 2^U \rightarrow \mathbb{R}$

Then f is called *monotone if* $f(S \cup \{a\}) \geq f(S)$. $S \subset U,\ a \in U$

Then f is called *sub-modular if*

$$f(S \cup \{a\}) - f(S) \geq f(T \cup \{a\}) - f(T),\ S \subseteq T.$$

Diminishing Returns Property: Value of adding an element

to smaller set is more than that of the bigger set

Examples

Entropy: Proof (conditioning reduces entropy)

Examples

Entropy: Proof (conditioning reduces entropy)

Take vector $A \subset B$ and element x not in B

Examples

Entropy: Proof (conditioning reduces entropy)

Take vector $A \subset B$ and element x not in B

To Show $\quad H(A \cup x) - H(A) >= H(B \cup x) - H(B)$

Examples

Entropy: Proof (conditioning reduces entropy)

Take vector $A \subset B$ and element x not in B

To Show $H(A \cup x) - H(A) >= H(B \cup x) - H(B)$

$H(x|A)$ $H(x|B)$

Examples

Entropy: Proof (conditioning reduces entropy)

Take vector $A \subset B$ and element x not in B

To Show $H(A \cup x) - H(A) >= H(B \cup x) - H(B)$

$$H(x|A) \quad >= \quad H(x|B)$$

conditioning

Examples

Entropy: Proof (conditioning reduces entropy)

Take vector $A \subset B$ and element x not in B

To Show $H(A \cup x) - H(A) >= H(B \cup x) - H(B)$

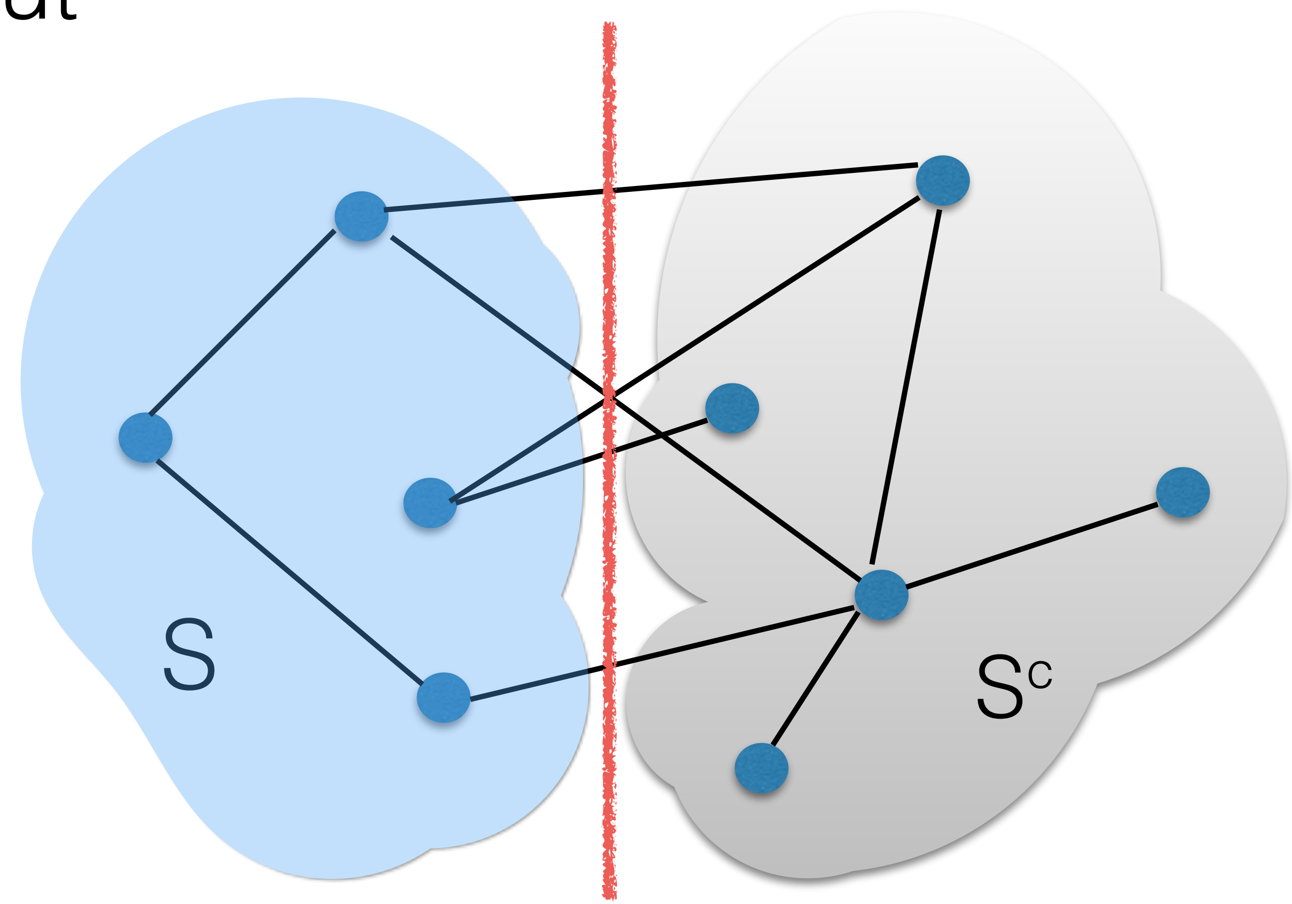
$H(x|A) >= H(x|B)$

conditioning

Mutual Information

Examples

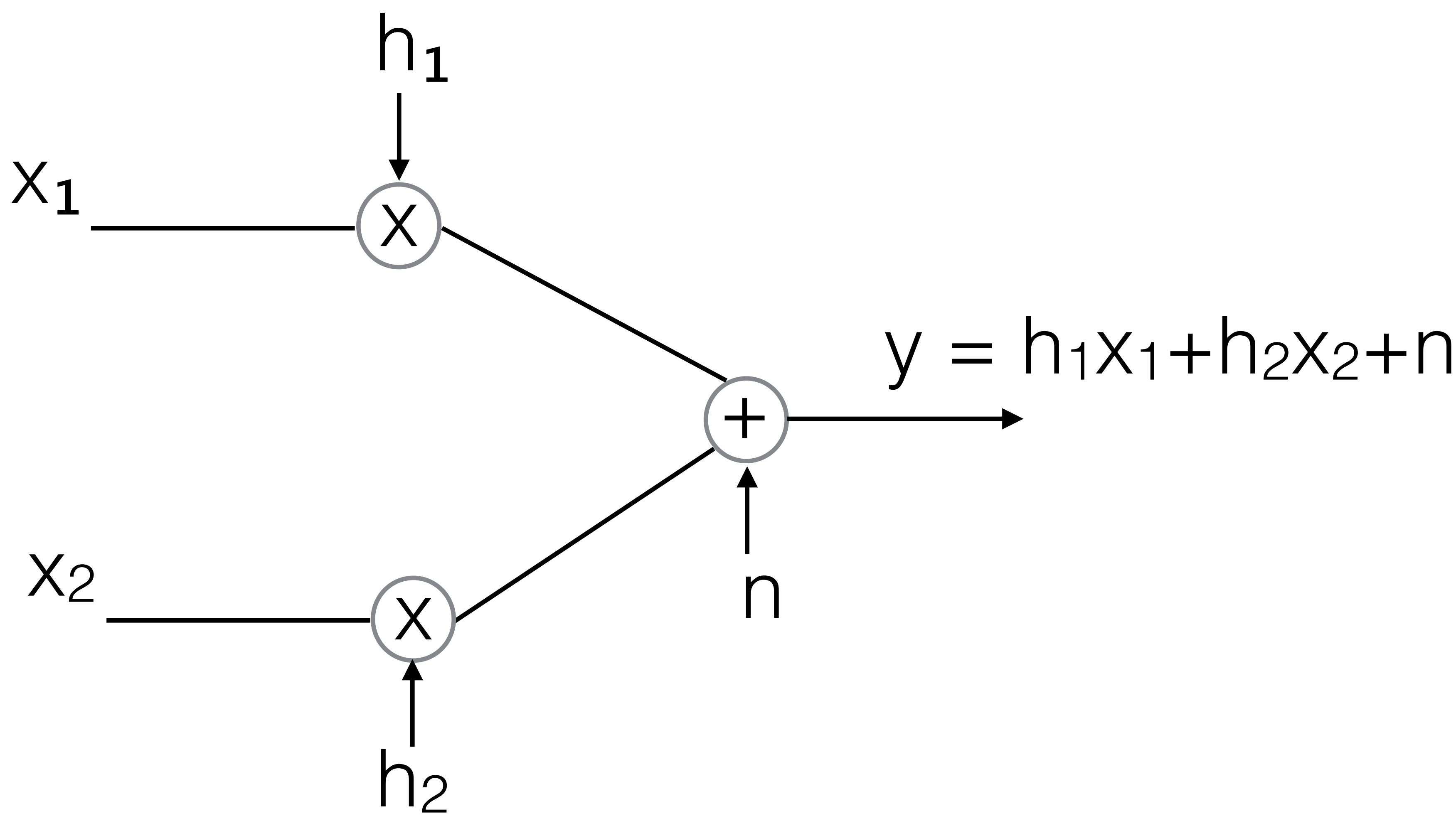
Graph Cut



# of edges crossing the cut (S, S^c)

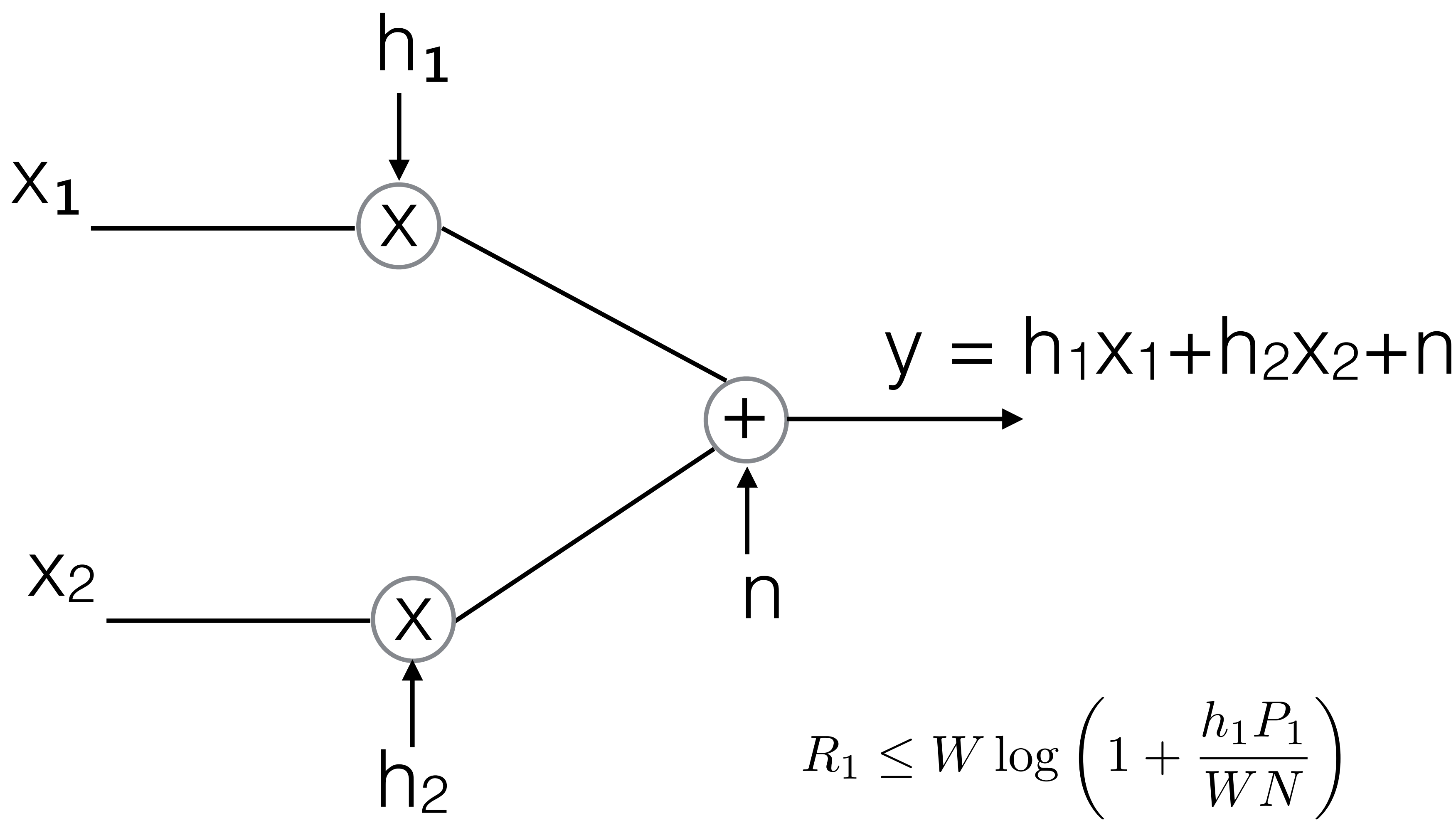
Multiple Access Channel (MAC)

Two-user



Multiple Access Channel (MAC)

Two-user

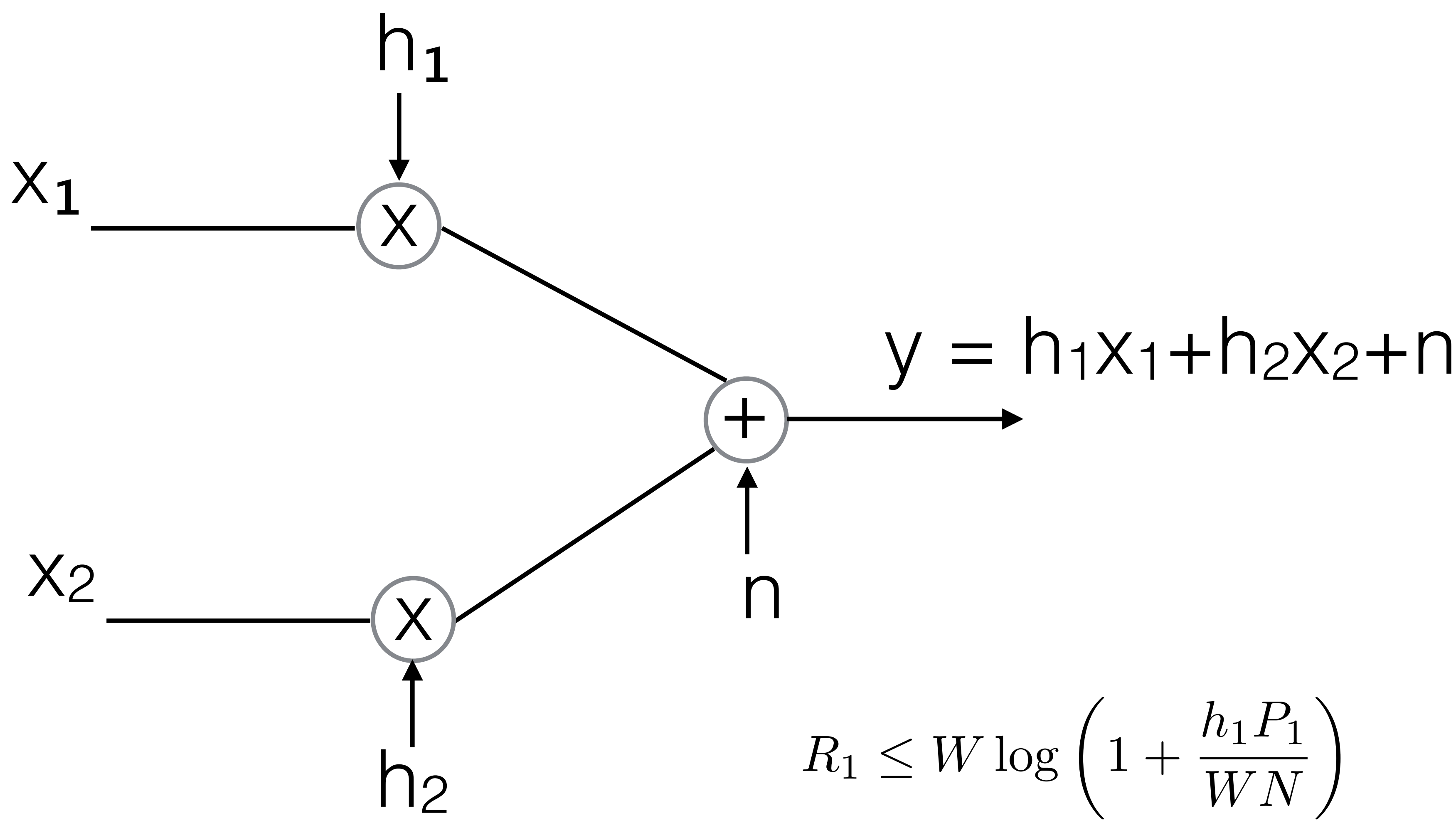


$$R_1 \leq W \log\left(1 + \frac{h_1 P_1}{WN}\right)$$

$$R_1 + R_2 \leq W \log\left(1 + \frac{h_1 P_1 + h_2 P_2}{WN}\right)$$

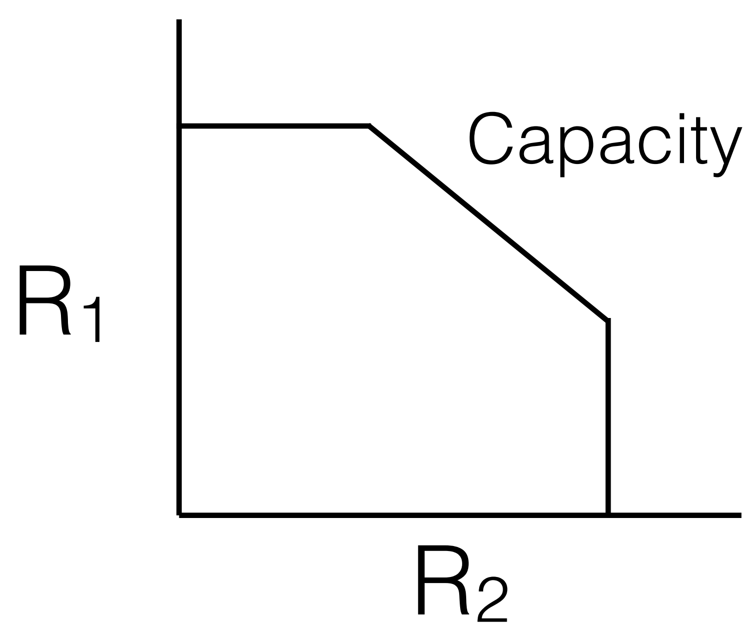
Multiple Access Channel (MAC)

Two-user



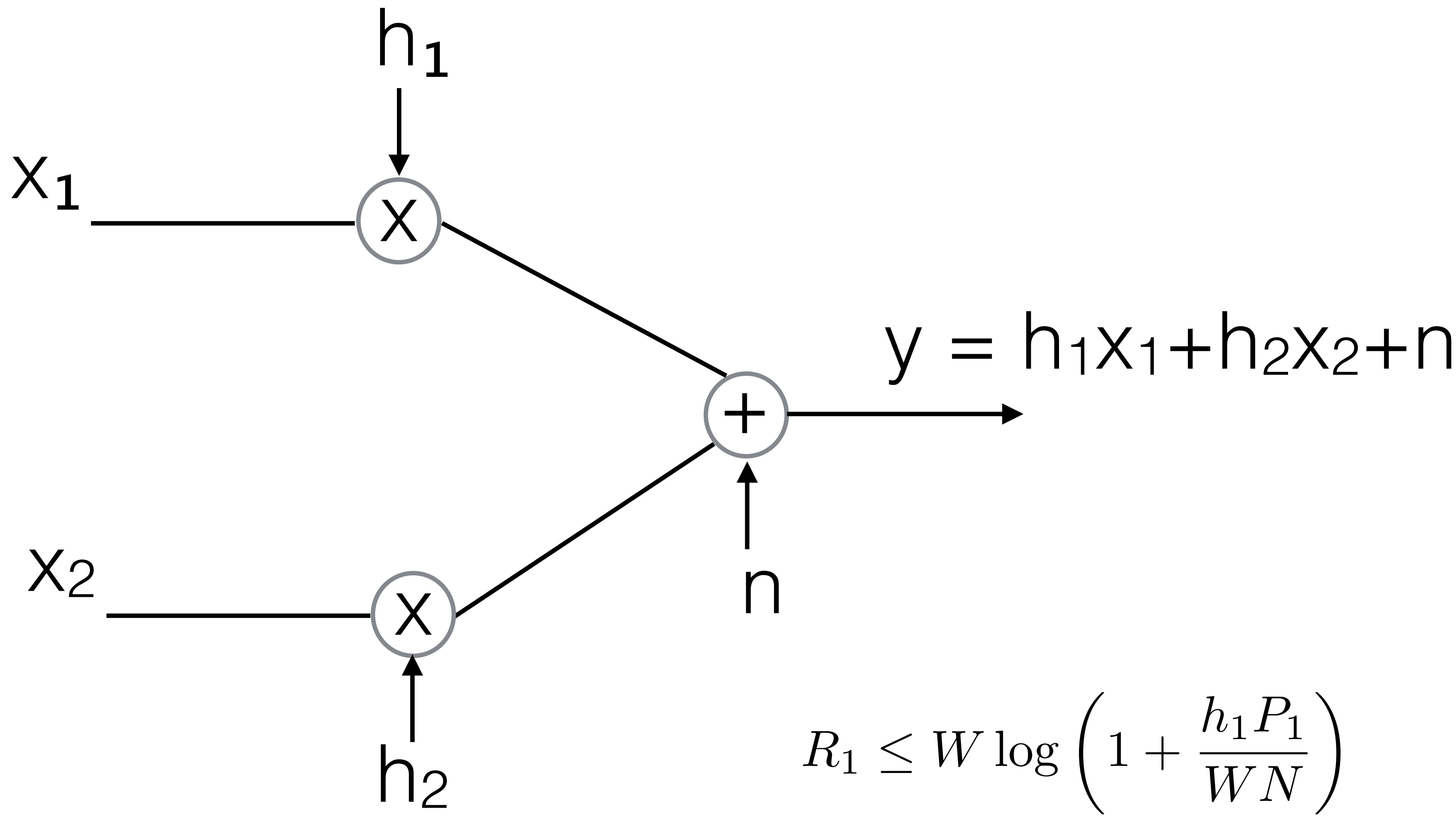
$$R_1 \leq W \log\left(1 + \frac{h_1 P_1}{WN}\right)$$

$$R_1 + R_2 \leq W \log\left(1 + \frac{h_1 P_1 + h_2 P_2}{WN}\right)$$



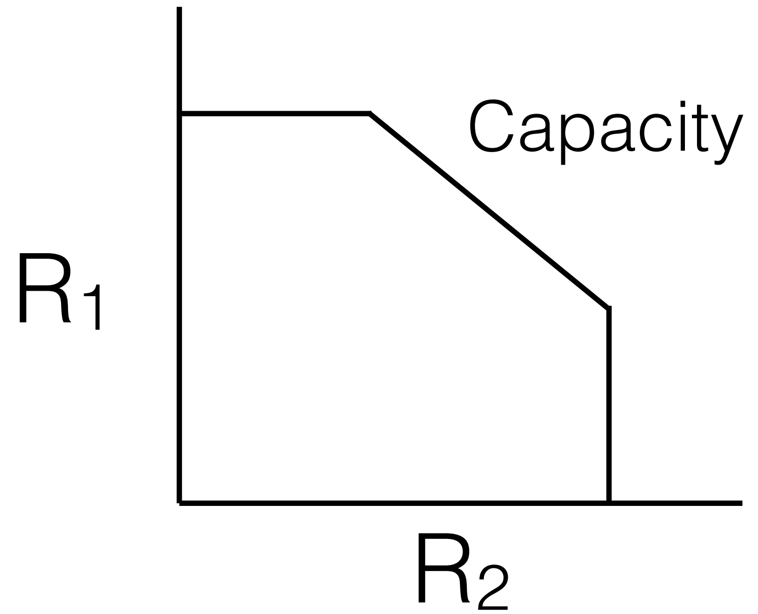
Multiple Access Channel (MAC)

Two-user



$$R_1 \leq W \log\left(1 + \frac{h_1 P_1}{WN}\right)$$

$$R_1 + R_2 \leq W \log\left(1 + \frac{h_1 P_1 + h_2 P_2}{WN}\right)$$



MAC requires Joint/Successive Decoding

FDMA

Instead use FDMA: non-overlapping frequency use

FDMA

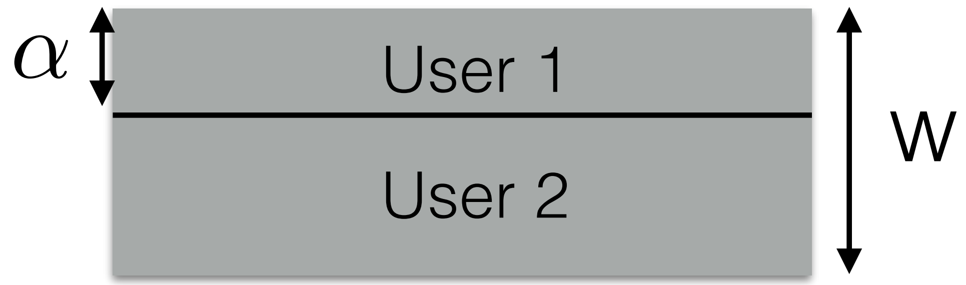
Instead use FDMA: non-overlapping frequency use

Simple signal processing

FDMA

Instead use FDMA: non-overlapping frequency use

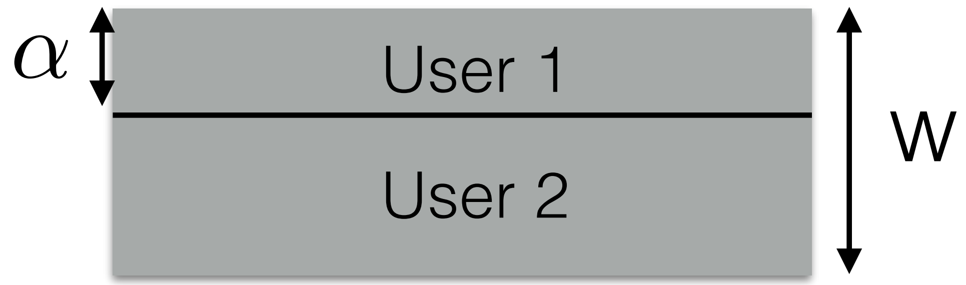
Simple signal processing



FDMA

Instead use FDMA: non-overlapping frequency use

Simple signal processing



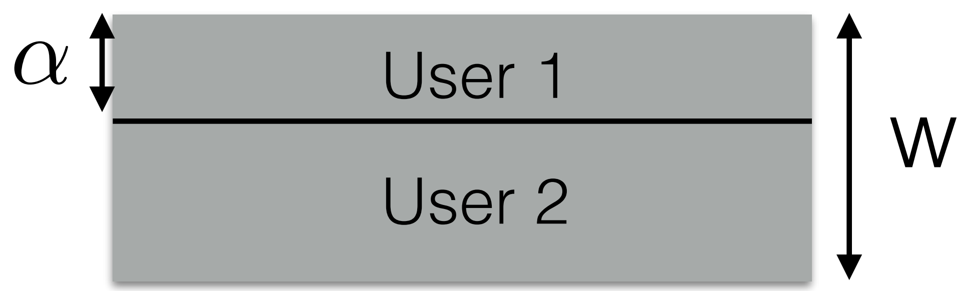
$$R_1 \leq \alpha W \log\left(1 + \frac{h_1 P_1}{\alpha W N}\right)$$

$$R_2 \leq (1-\alpha) W \log\left(1 + \frac{h_2 P_2}{(1-\alpha) W N}\right)$$

FDMA

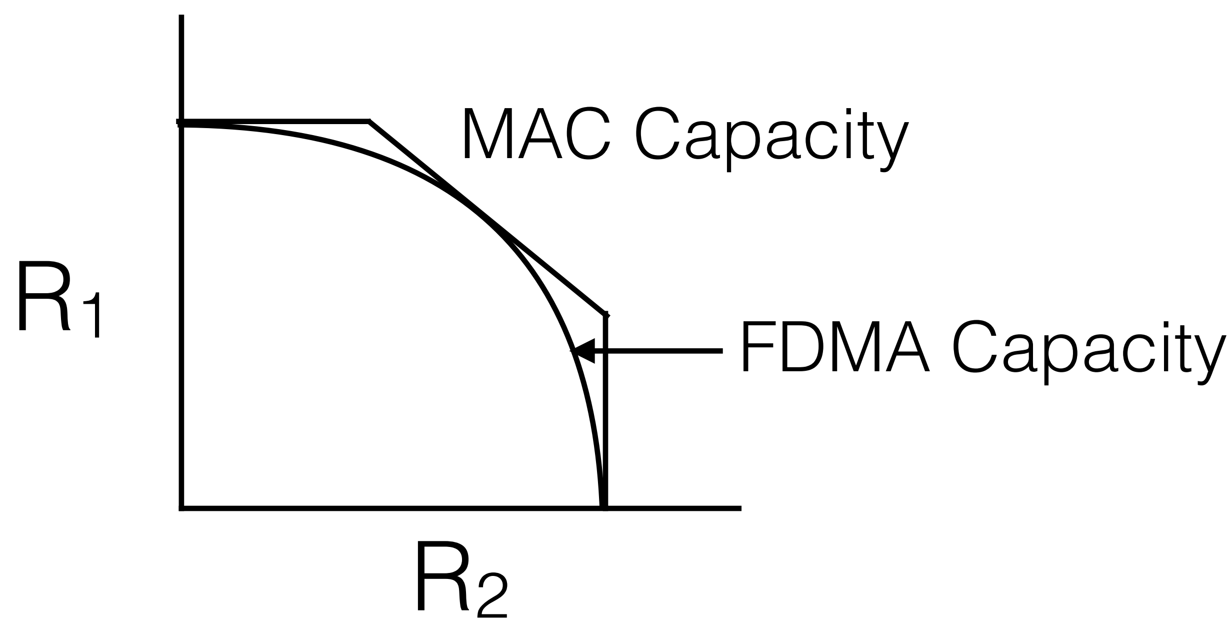
Instead use FDMA: non-overlapping frequency use

Simple signal processing



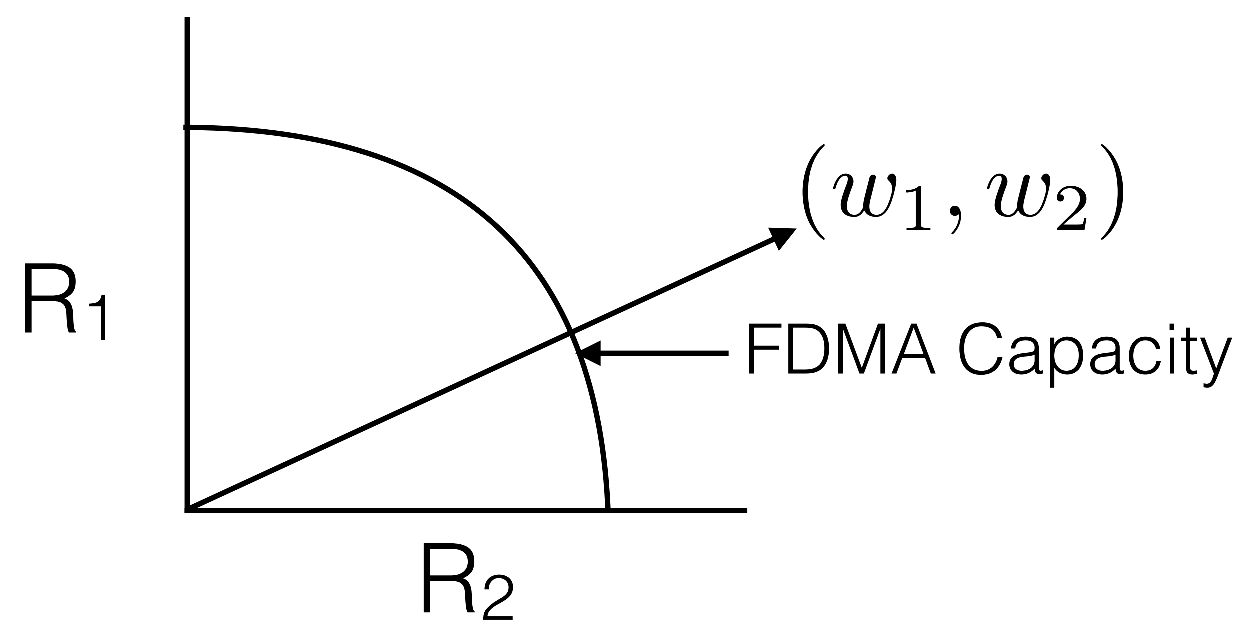
$$R_1 \leq \alpha W \log\left(1 + \frac{h_1 P_1}{\alpha W N}\right)$$

$$R_2 \leq (1-\alpha) W \log\left(1 + \frac{h_2 P_2}{(1-\alpha) W N}\right)$$



Capacity Region of n Users FDMA

For each vector $(w_1, w_2, \ldots, w_n)$



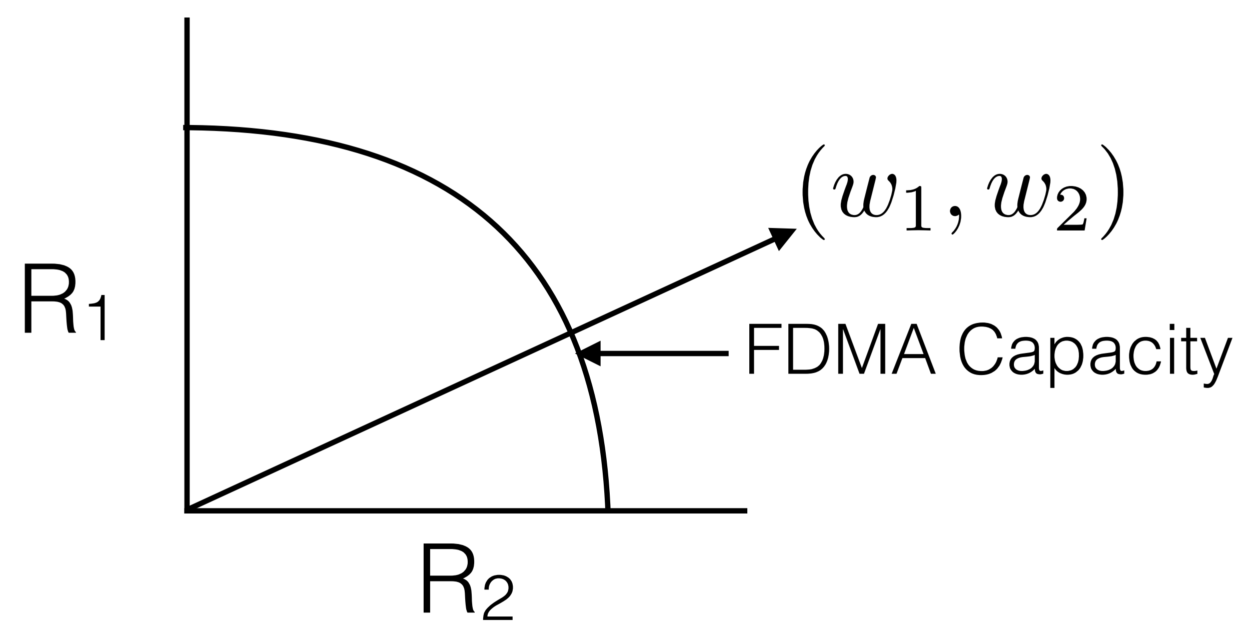
Capacity Region of n Users FDMA

For each vector $(w_1, w_2, \ldots, w_n)$

Find opt power $\max_{P_i(f)} \sum_{i=1}^{n} w_i R_i$

non-overlapping freq $P_i(f)P_j(f) = 0$

$f \in W$



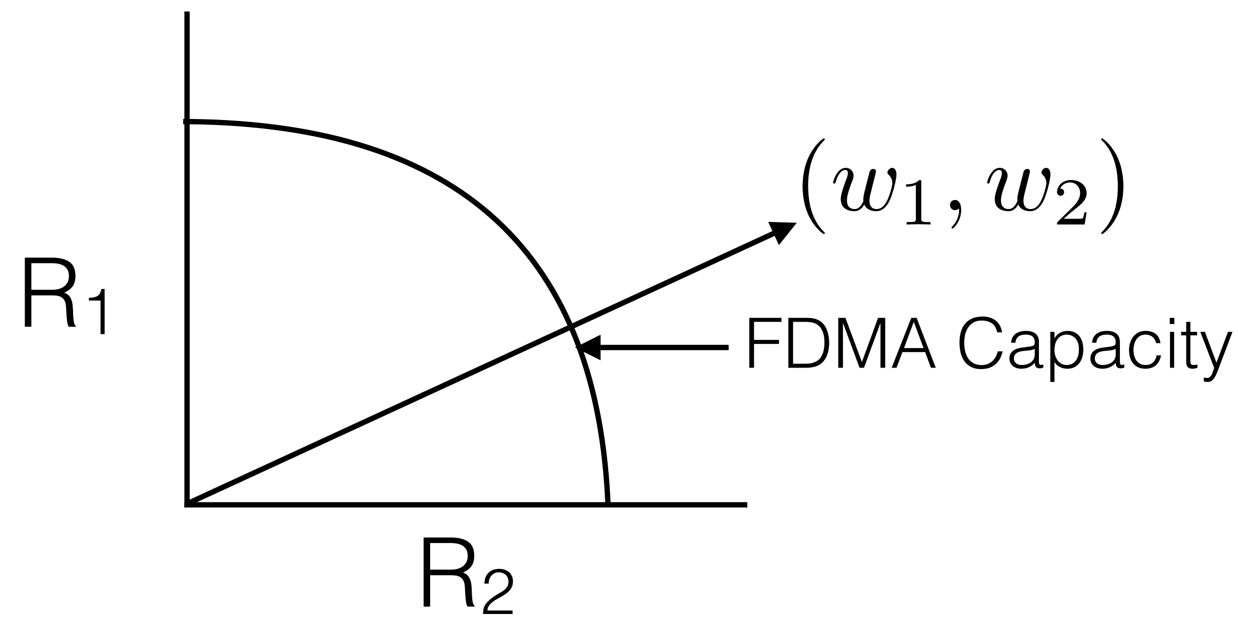
Capacity Region of n Users FDMA

For each vector $(w_1, w_2, \ldots, w_n)$

Find opt power $$\max_{P_i(f)} \sum_{i=1}^{n} w_i R_i$$

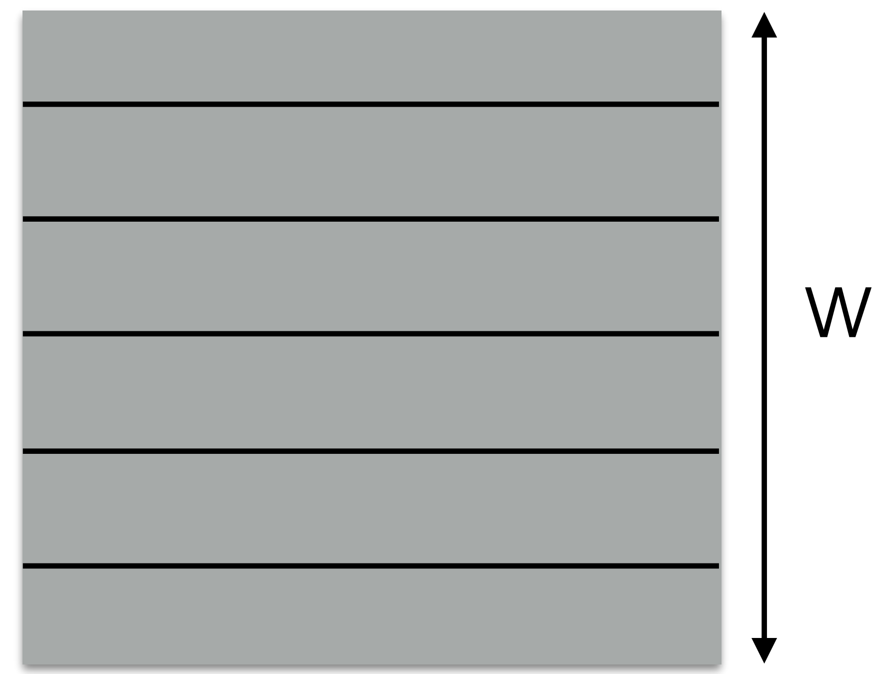
non-overlapping freq $$P_i(f) P_j(f) = 0$$

$$f \in W$$



Capacity Region of n Users Discretized FDMA

BW W is partitioned into m bins

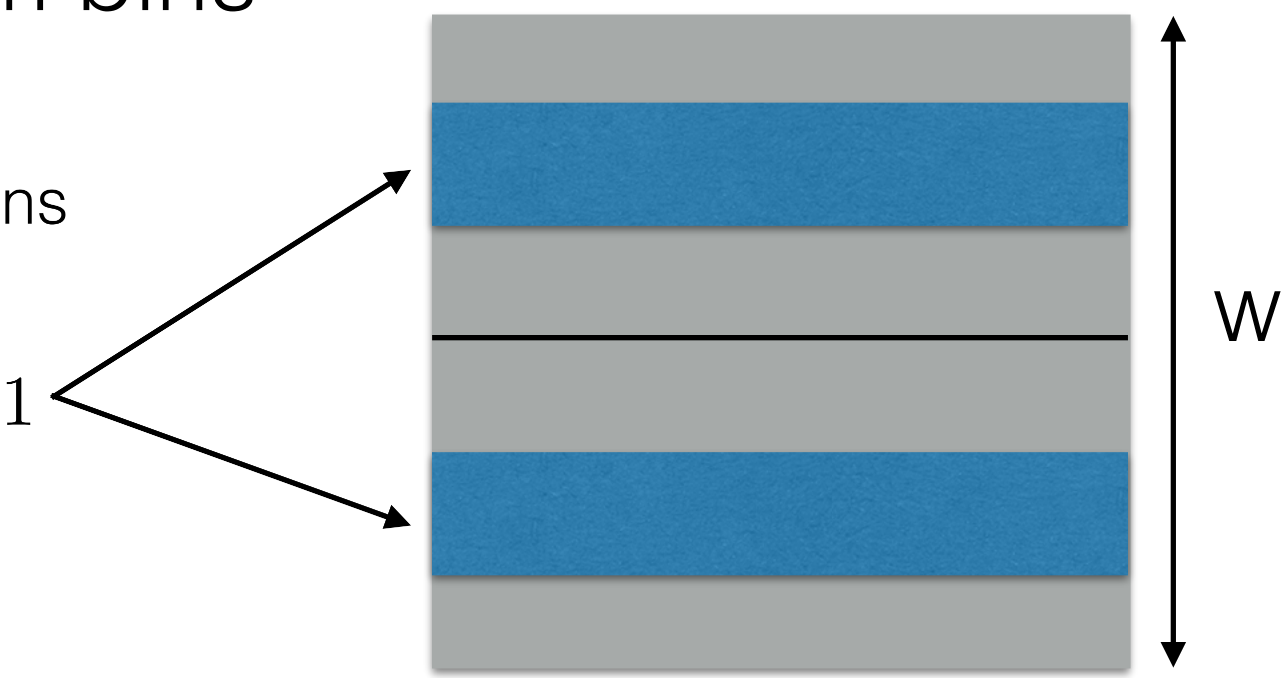


Capacity Region of n Users Discretized FDMA

BW W is partitioned into m bins

Each user is allocated one or more bins

User i is allocated bin j if $b_{ij} = 1$

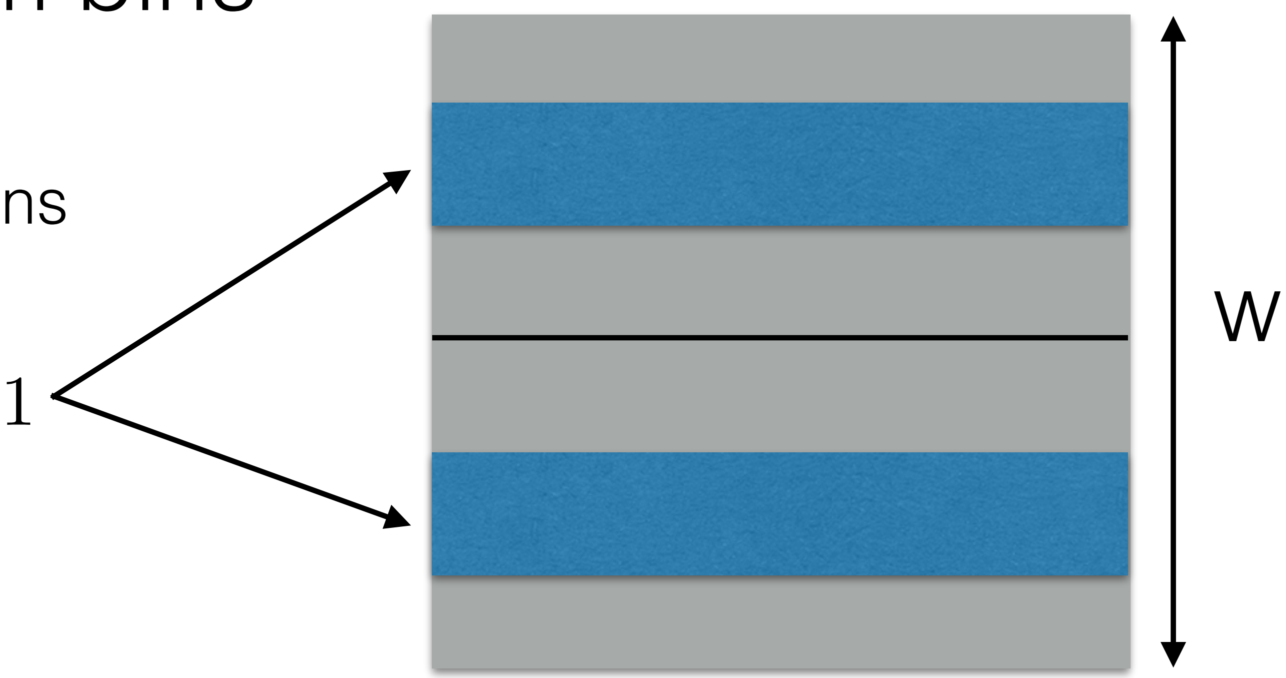


Capacity Region of n Users Discretized FDMA

BW W is partitioned into m bins

Each user is allocated one or more bins

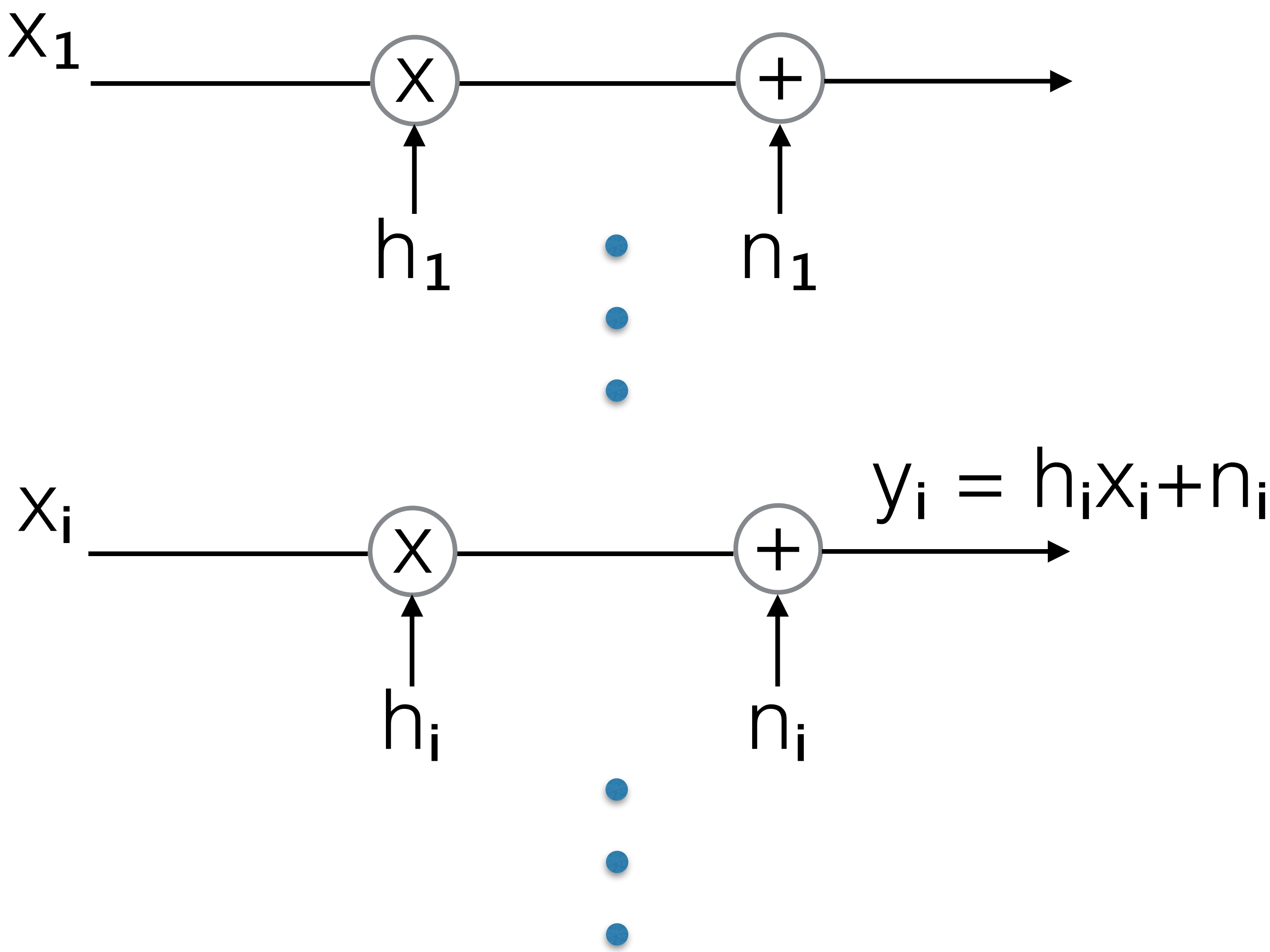
User i is allocated bin j if $b_{ij} = 1$



Find bin and power allocation
for each vector $(w_1, w_2, \ldots, w_n)$

$$\max_{P_i, b_{ij} \in \{0,1\}} \sum_{i=1}^{n} w_i \sum_{j=1}^{F_j} b_{ij} R_{ij}$$

Each user sees a Parallel Gaussian Channel



$$r_i = \log\left(1 + \frac{h_i}{n_i} P_i\right)$$

Find powers P_i's to max sum-rate

$$\max_{\sum_{i=1}^{n} P_i \leq P} \sum_{i=1}^{n} \log\left(1 + \frac{h_i}{n_i} P_i\right)$$

Water-filling

Optimal Solution to $\max_{\sum_{i=1}^n P_i \leq P} \sum_{i=1}^n \log\left(1 + \frac{h_i}{n_i} P_i\right)$

is Water-filling

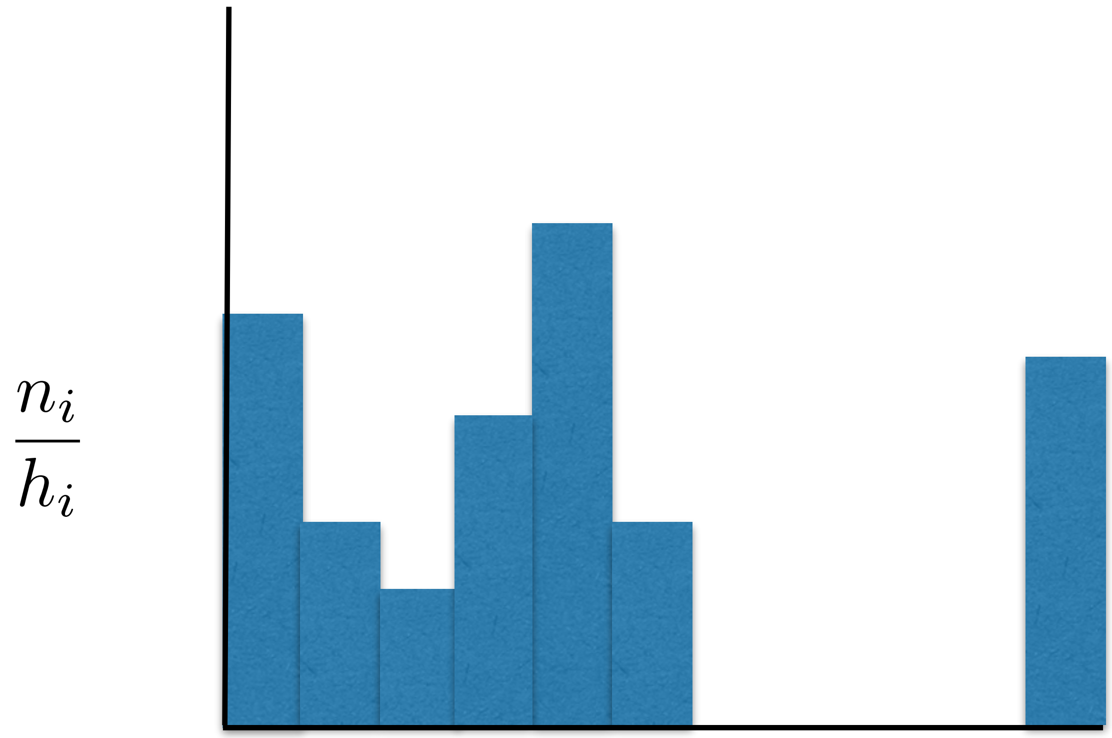
$$P_i^\star = \left(\mu - \frac{n_i}{h_i}\right)^+ \qquad \sum_{i=1}^n P_i^\star = P$$

Water-filling

Optimal Solution to $\max_{\sum_{i=1}^n P_i \leq P} \sum_{i=1}^n \log\left(1 + \frac{h_i}{n_i} P_i\right)$

is Water-filling

$$P_i^\star = \left(\mu - \frac{n_i}{h_i}\right)^+ \qquad \sum_{i=1}^n P_i^\star = P$$

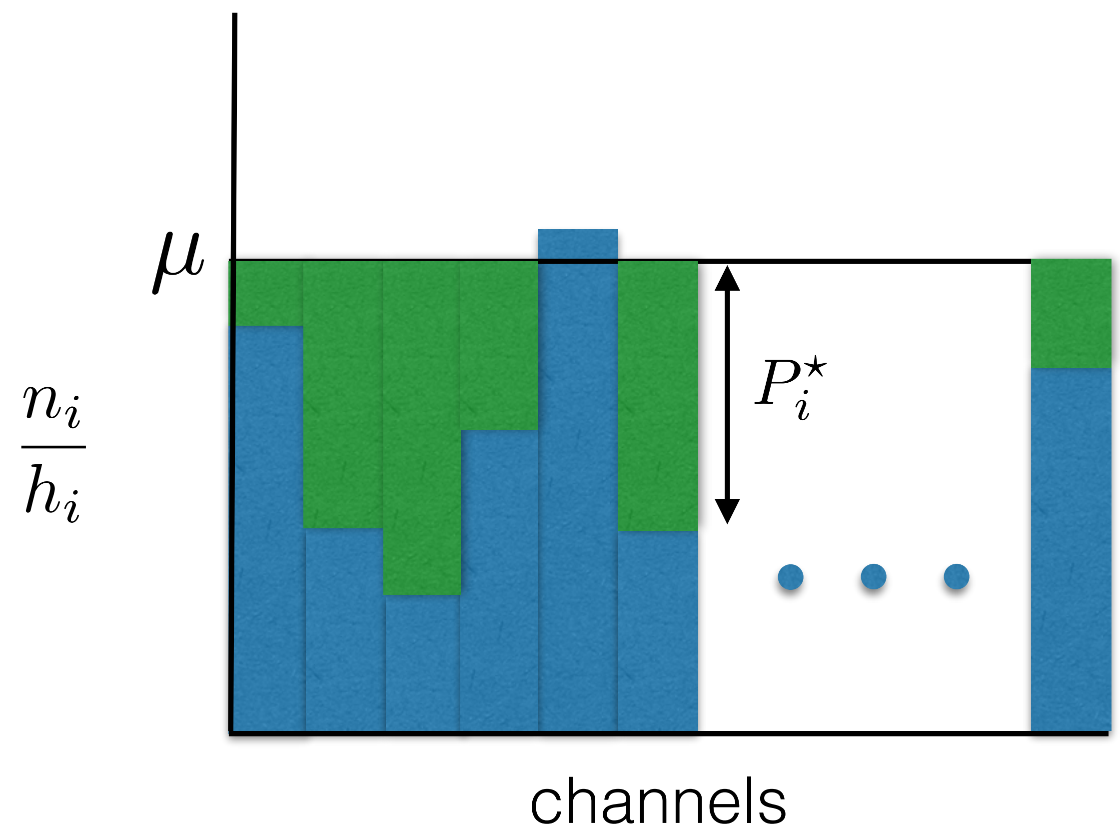


Water-filling

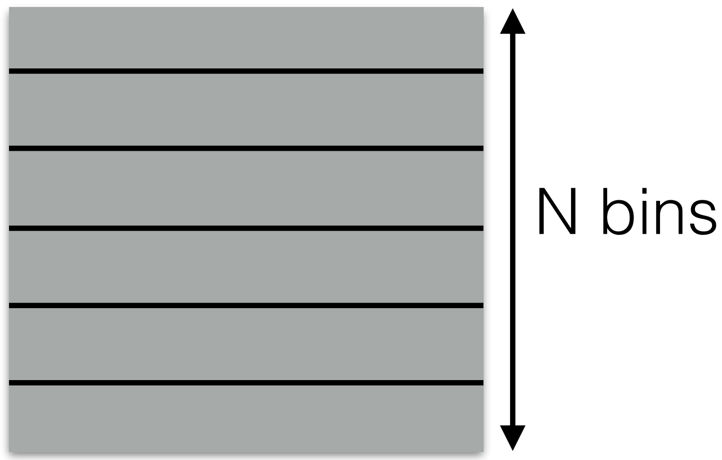
Optimal Solution to $\max_{\sum_{i=1}^n P_i \leq P} \sum_{i=1}^n \log\left(1 + \frac{h_i}{n_i} P_i\right)$

is Water-filling

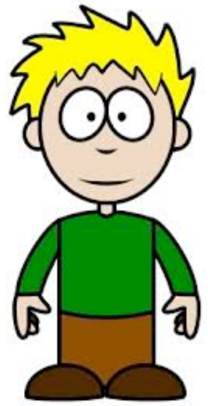
$$P_i^\star = \left(\mu - \frac{n_i}{h_i}\right)^+ \qquad \sum_{i=1}^n P_i^\star = P$$

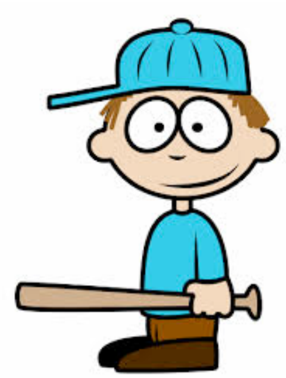


OFDMA Capacity as Partitioning Problem

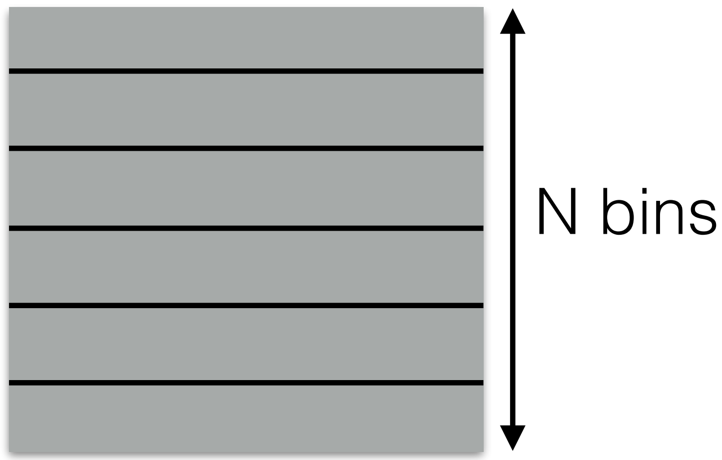


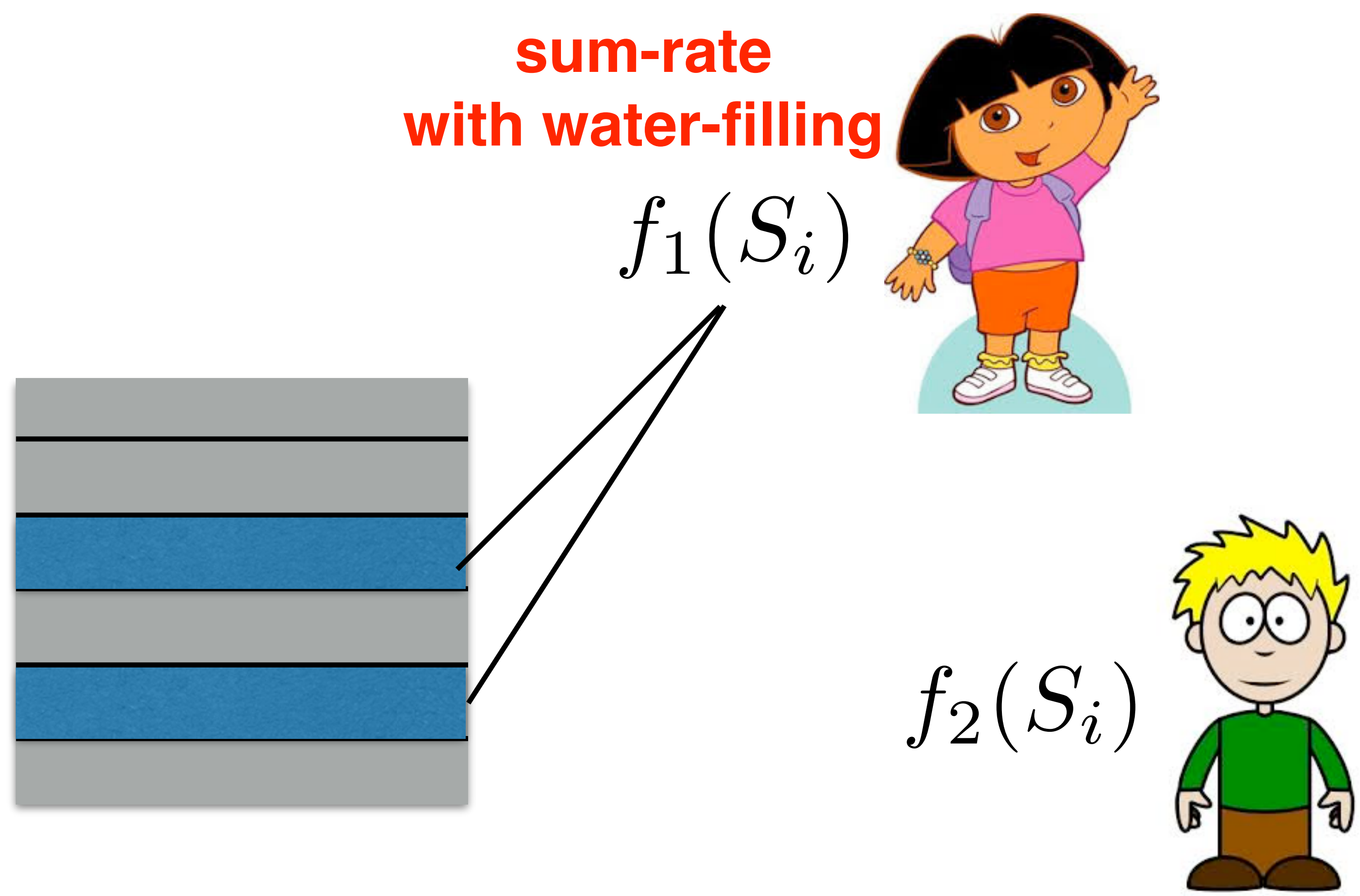
sum-rate with water-filling





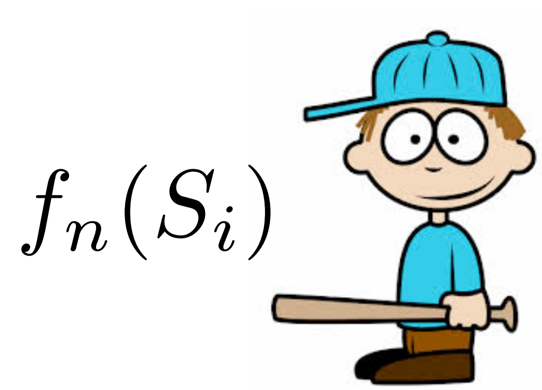
OFDMA Capacity as Partitioning Problem





Partition N bins

into $S_1, S_2, \ldots, S_n$



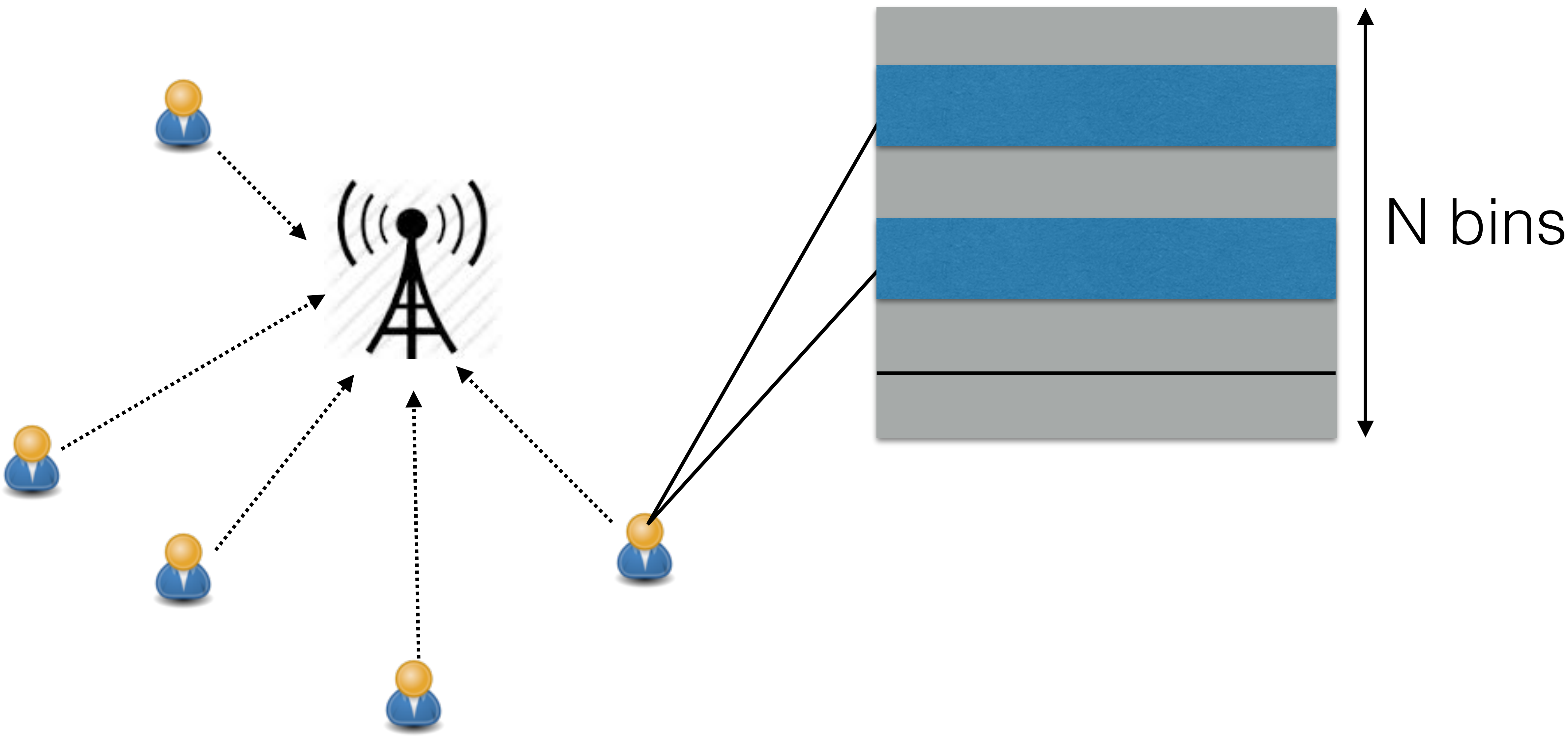
Known Results

Heuristics

- Convex Relaxation (YuCioffi'02)
- KKT (KimHanKim'05)

Subcarrier and Power allocation in uplink OFDMA

Identical to finding OFDMA capacity



Given sub-carrier (bins) allocation, find power using water-filling to max sum-rate

1/2 Approx.

Use Greedy Algorithm

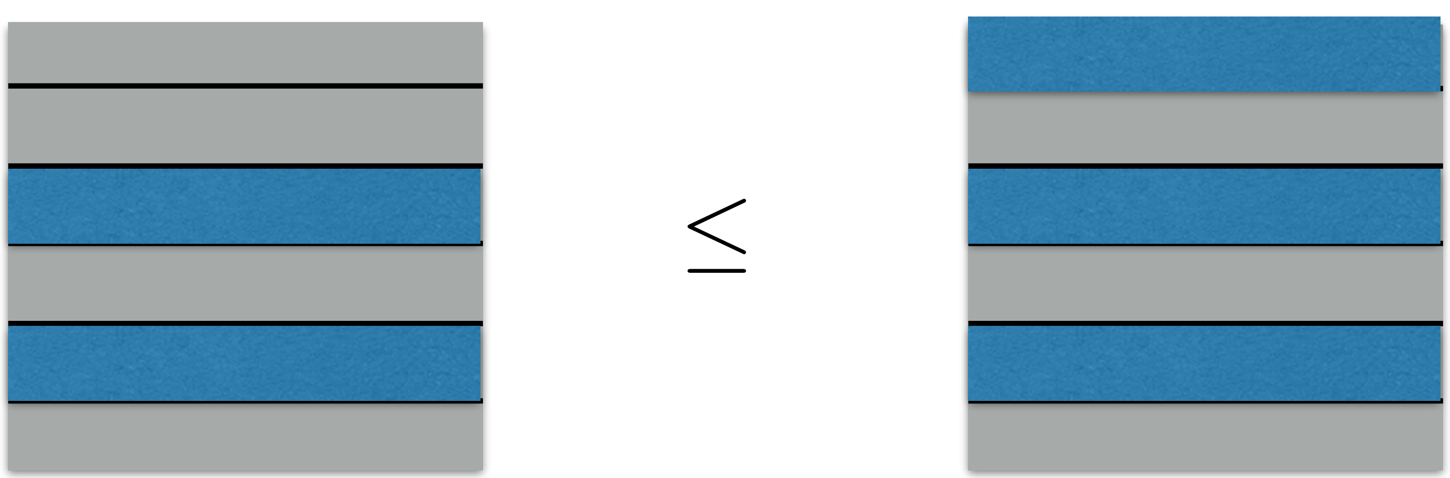
- check if each users incentive $\mathbf{f}_i$ (sum-rate with water-filling) is
  - sub-modular
  - monotone

1/2 Approx.

Use Greedy Algorithm

- check if each users incentive $\mathbf{f}_i$ (sum-rate with water-filling) is
  - sub-modular
  - monotone

Monotonicity is clear : More channels give larger rate

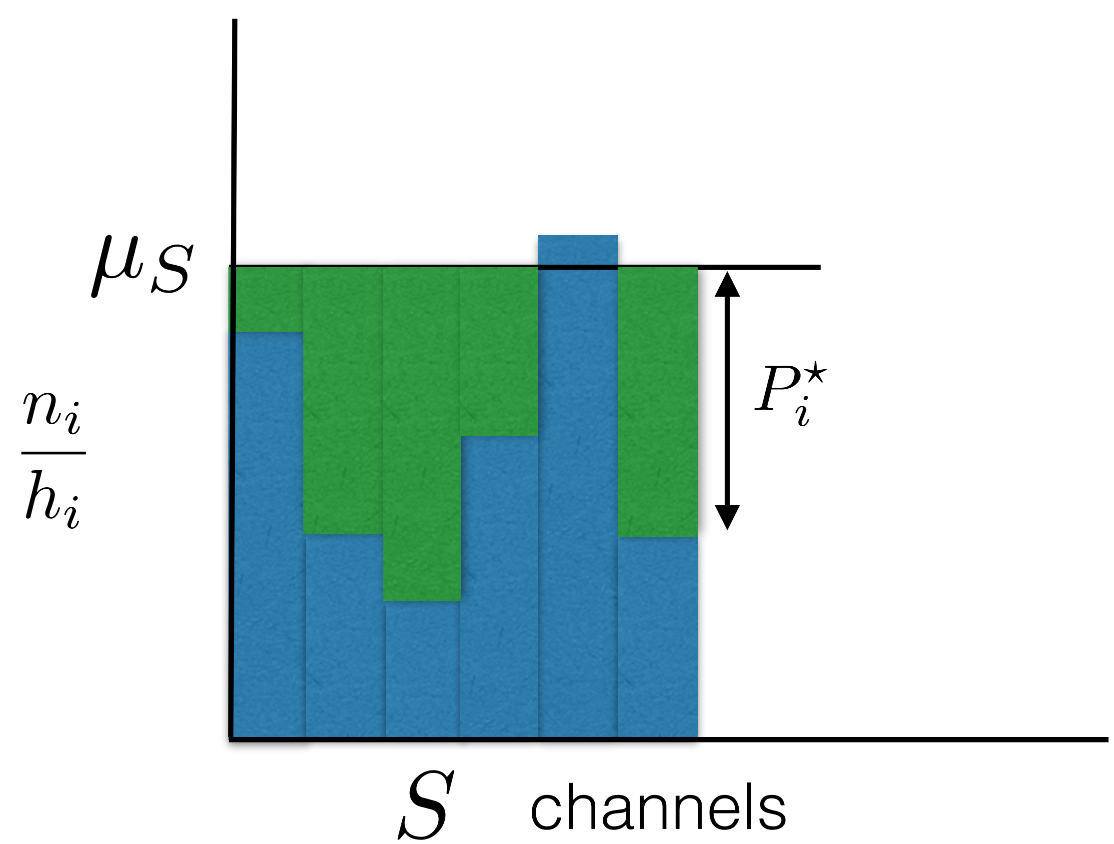


Water-filling as a Set-Function

Let S be a set of channels then sum-rate is

$$R(S) = \max_{\sum_{i=1}^{n} P_i \leq P} \sum_{i \in S} \log\left(1 + \frac{h_i}{n_i} P_i\right)$$

Water-filling $\quad P_i^\star = \left(\mu_S - \frac{n_i}{h_i}\right) \qquad \sum_{i \in S} P_i^\star = P$

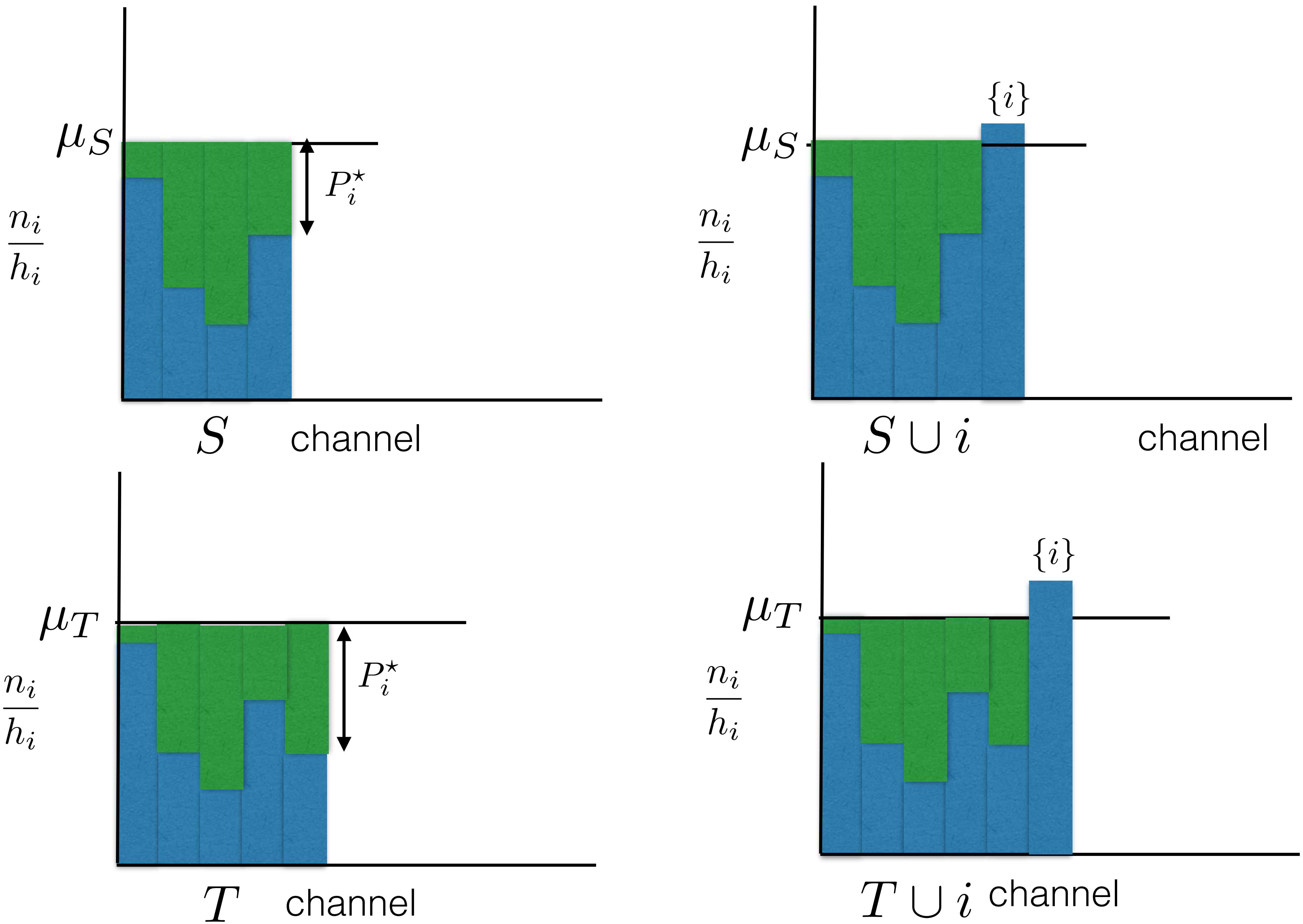


Sub-Modularity of Water-Filling

To check $\quad R(S \cup \{i\}) - R(S) \geq R(T \cup \{i\}) - R(T)$

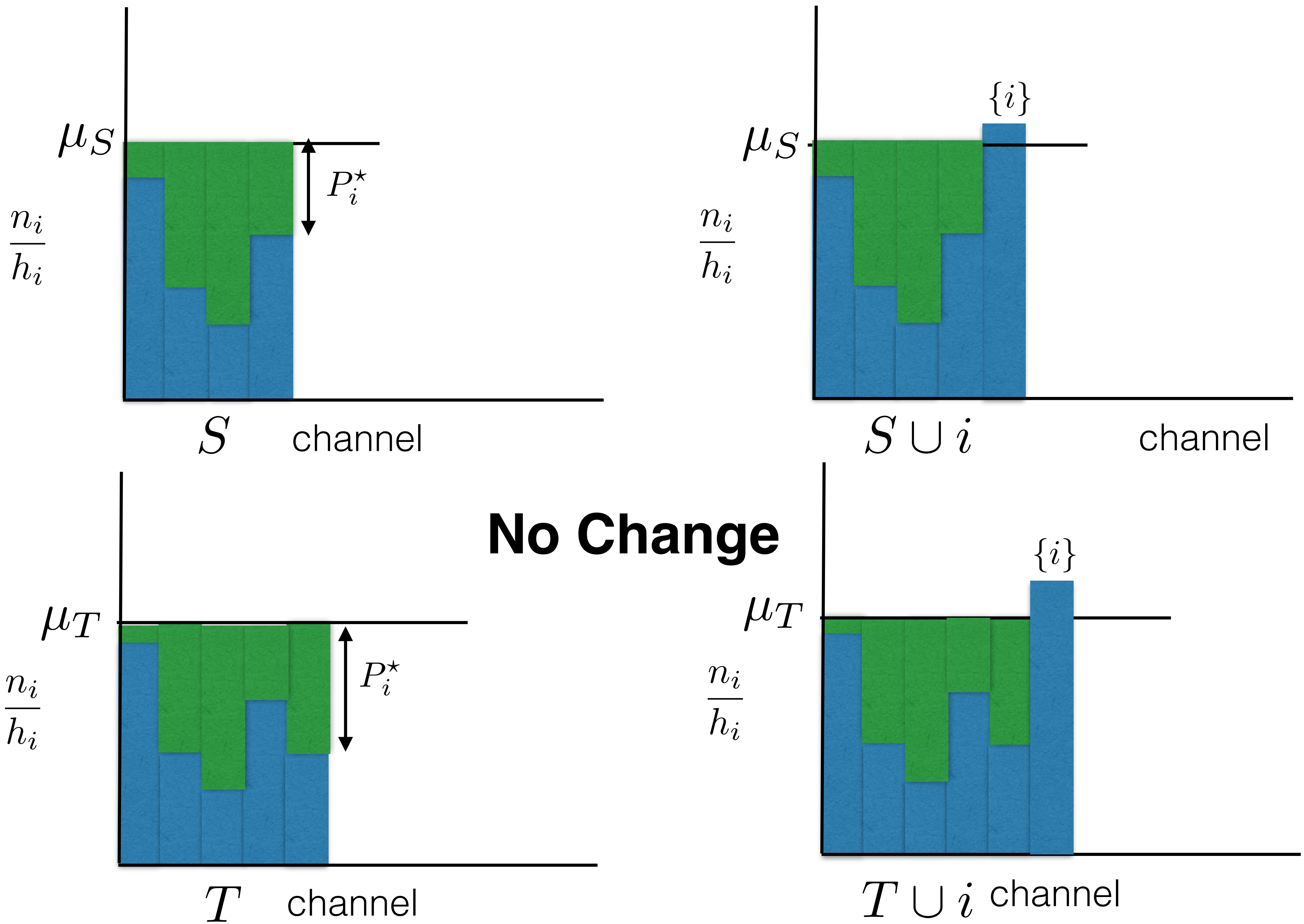
Sub-Modularity of Water-Filling

To check $R(S \cup \{i\}) - R(S) \geq R(T \cup \{i\}) - R(T)$



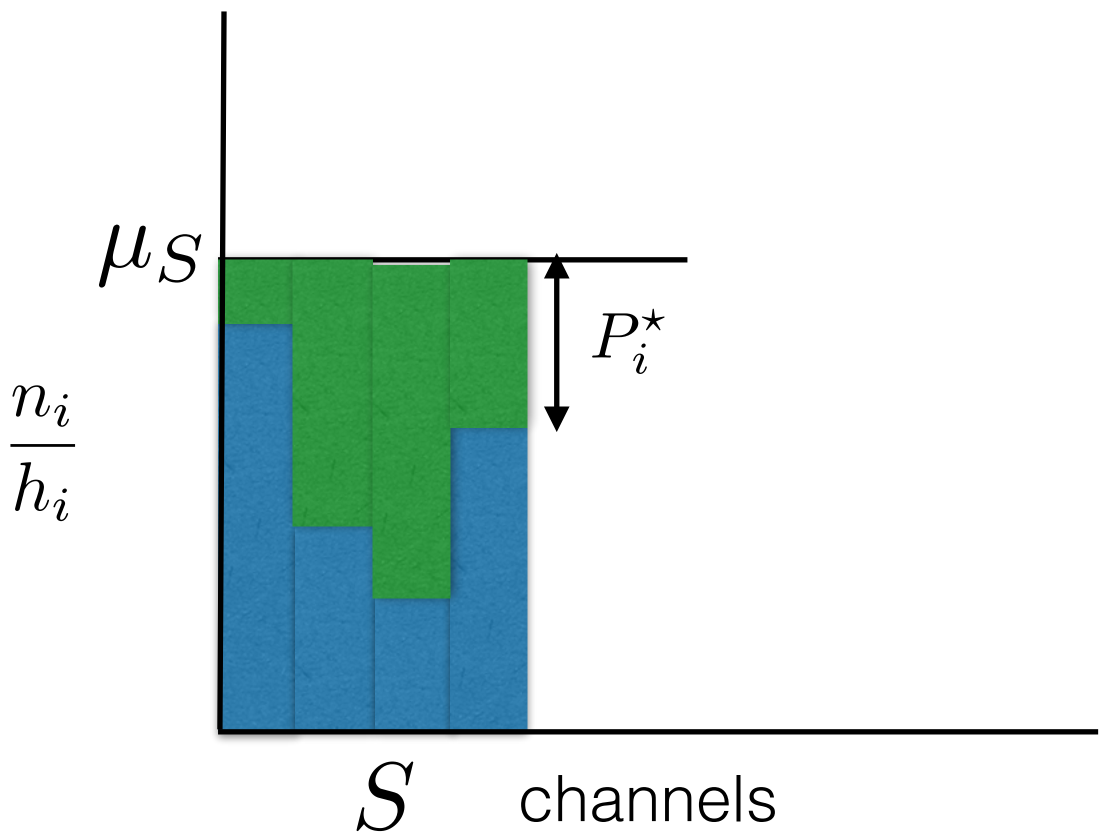
Sub-Modularity of Water-Filling

To check $R(S \cup \{i\}) - R(S) \geq R(T \cup \{i\}) - R(T)$

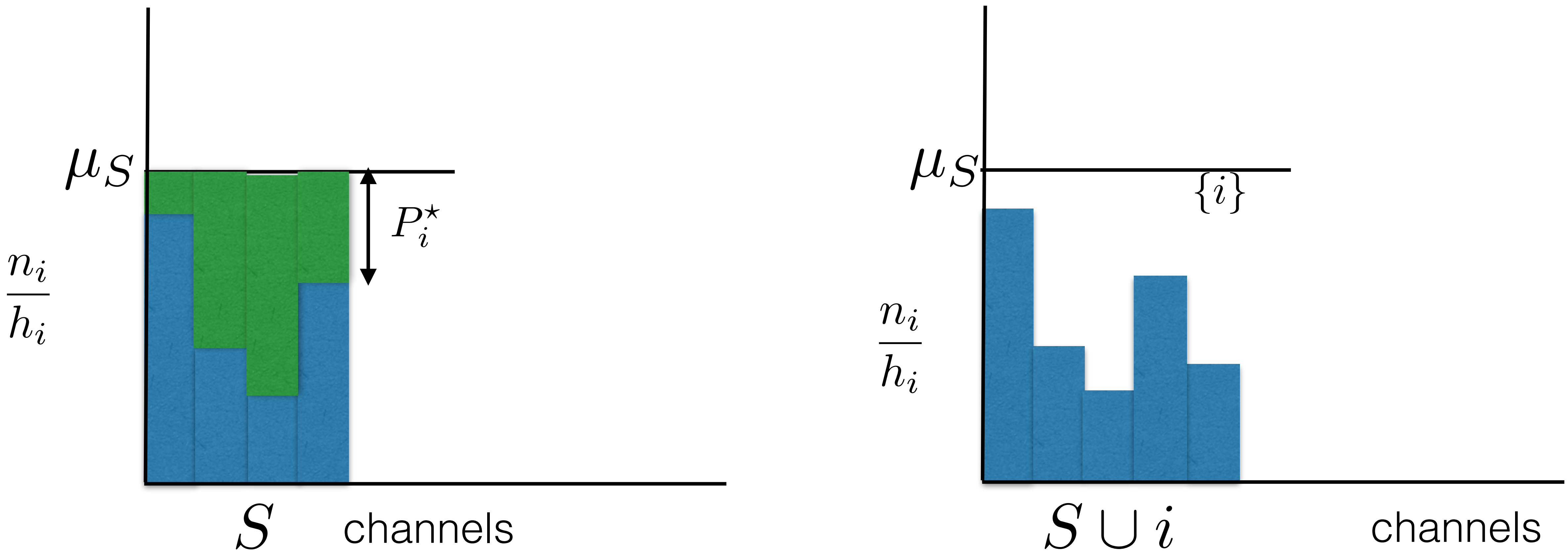


In general there is no easy way

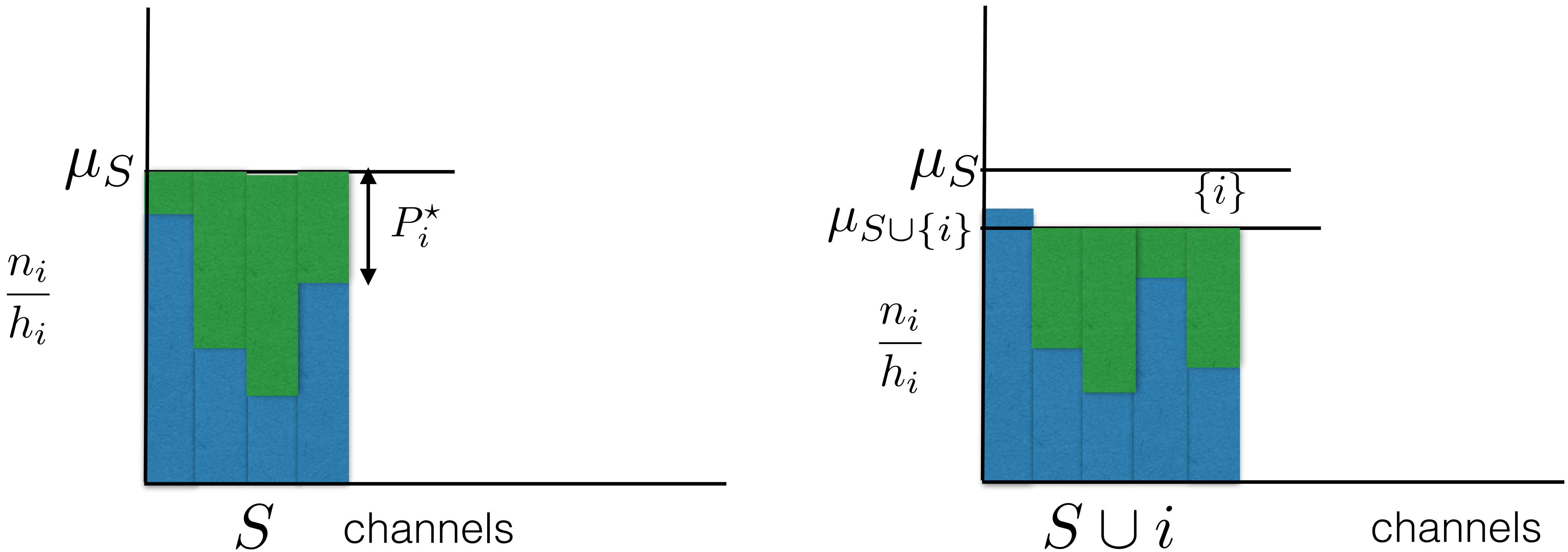
In general there is no easy way



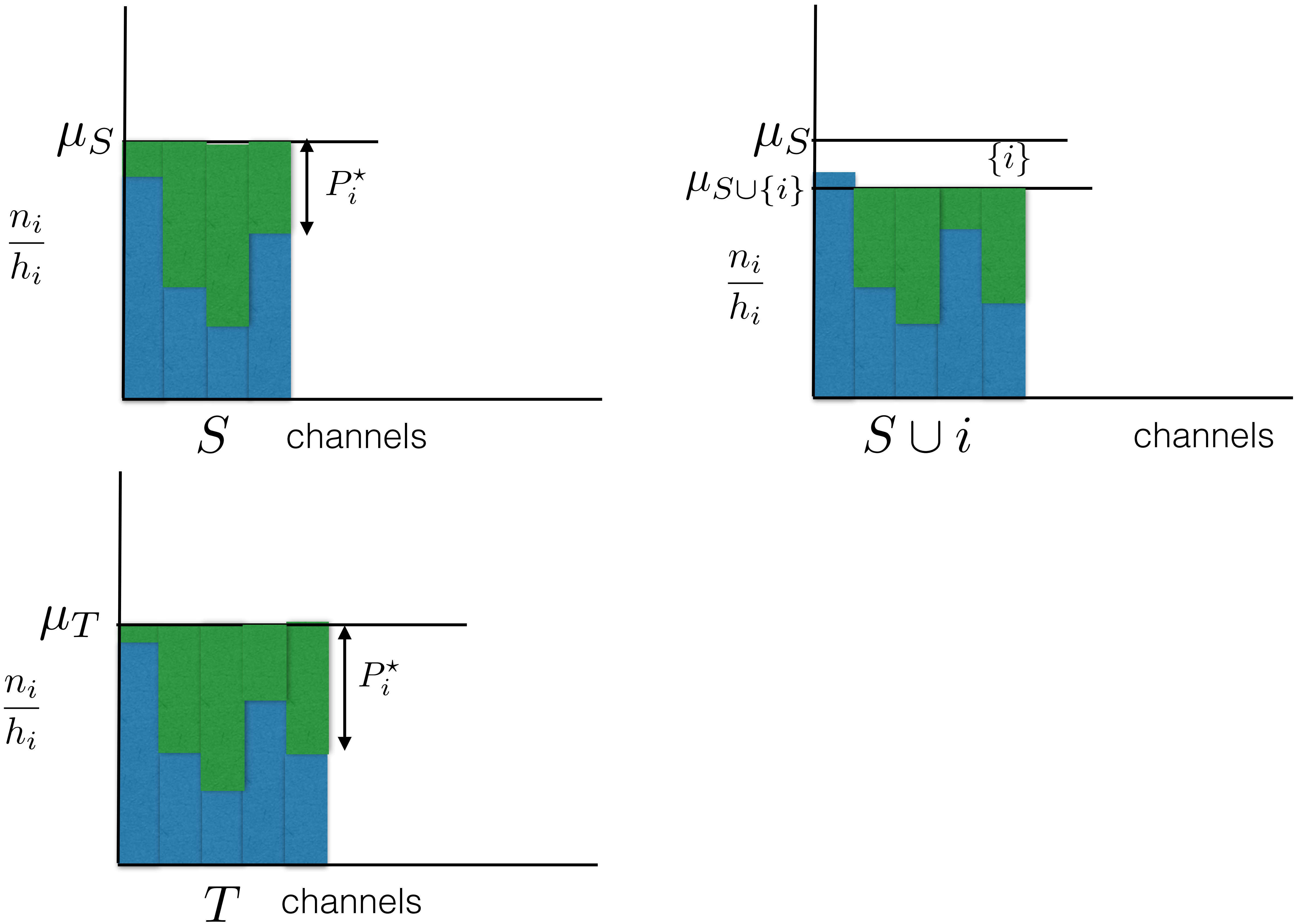
In general there is no easy way



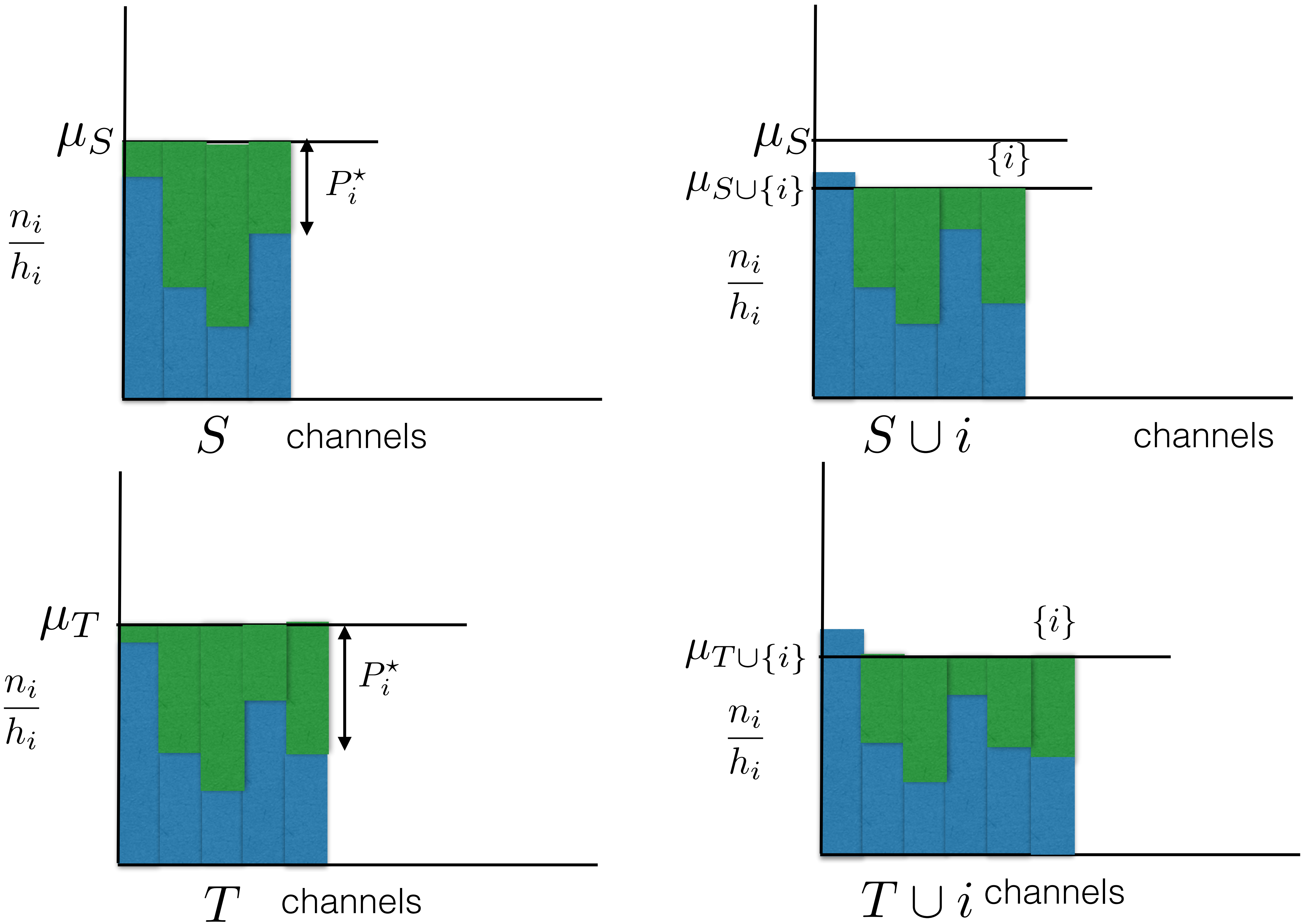
In general there is no easy way



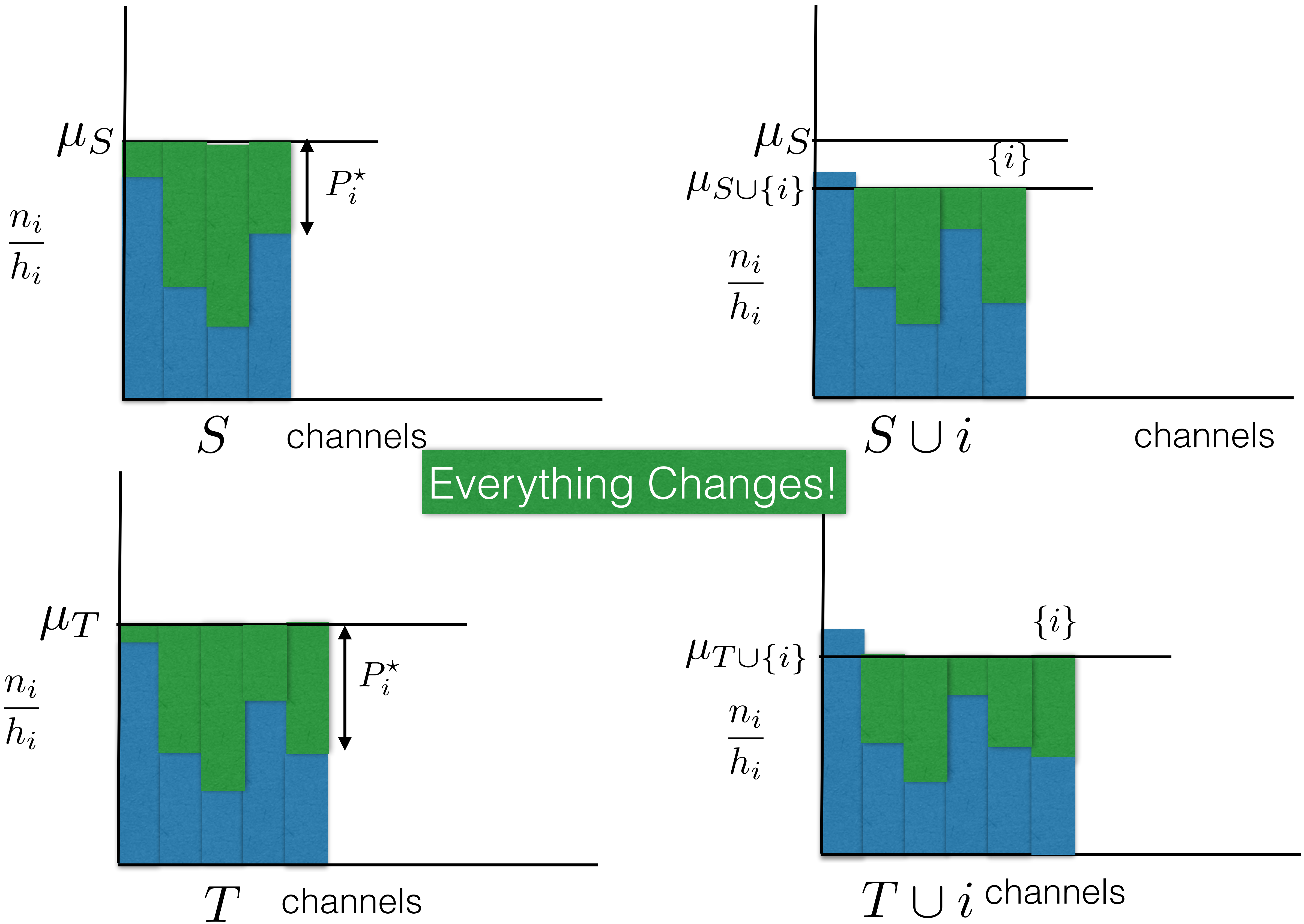
In general there is no easy way



In general there is no easy way



In general there is no easy way



Idea

A. Majorization

B. Karamata's Inequality

Idea

A. Majorization

B. Karamata's Inequality

Two vectors **a** and **b** arranged in descending order

$$\mathbf{a} = [a_1 a_2 \ldots a_n] \qquad \mathbf{b} = [b_1 b_2 \ldots b_n]$$

Idea

A. Majorization

B. Karamata's Inequality

Two vectors **a** and **b** arranged in descending order

$$\mathbf{a} = [a_1 a_2 \ldots a_n] \qquad \mathbf{b} = [b_1 b_2 \ldots b_n]$$

a majorizes **b** if $$\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} b_i, \forall\ k \leq n$$

Idea

A. Majorization

B. Karamata's Inequality

Two vectors **a** and **b** arranged in descending order

$$\mathbf{a} = [a_1 a_2 \ldots a_n] \qquad \mathbf{b} = [b_1 b_2 \ldots b_n]$$

a majorizes **b** if $\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} b_i, \forall\ k \leq n$

Karamata's inequality $\sum_{i=1}^{n} g(a_i) \geq \sum_{i=1}^{n} g(b_i)$ for any *convex* function g

Idea

A. Majorization

B. Karamata's Inequality

Two vectors **a** and **b** arranged in descending order

$$\mathbf{a} = [a_1 a_2 \ldots a_n] \qquad \mathbf{b} = [b_1 b_2 \ldots b_n]$$

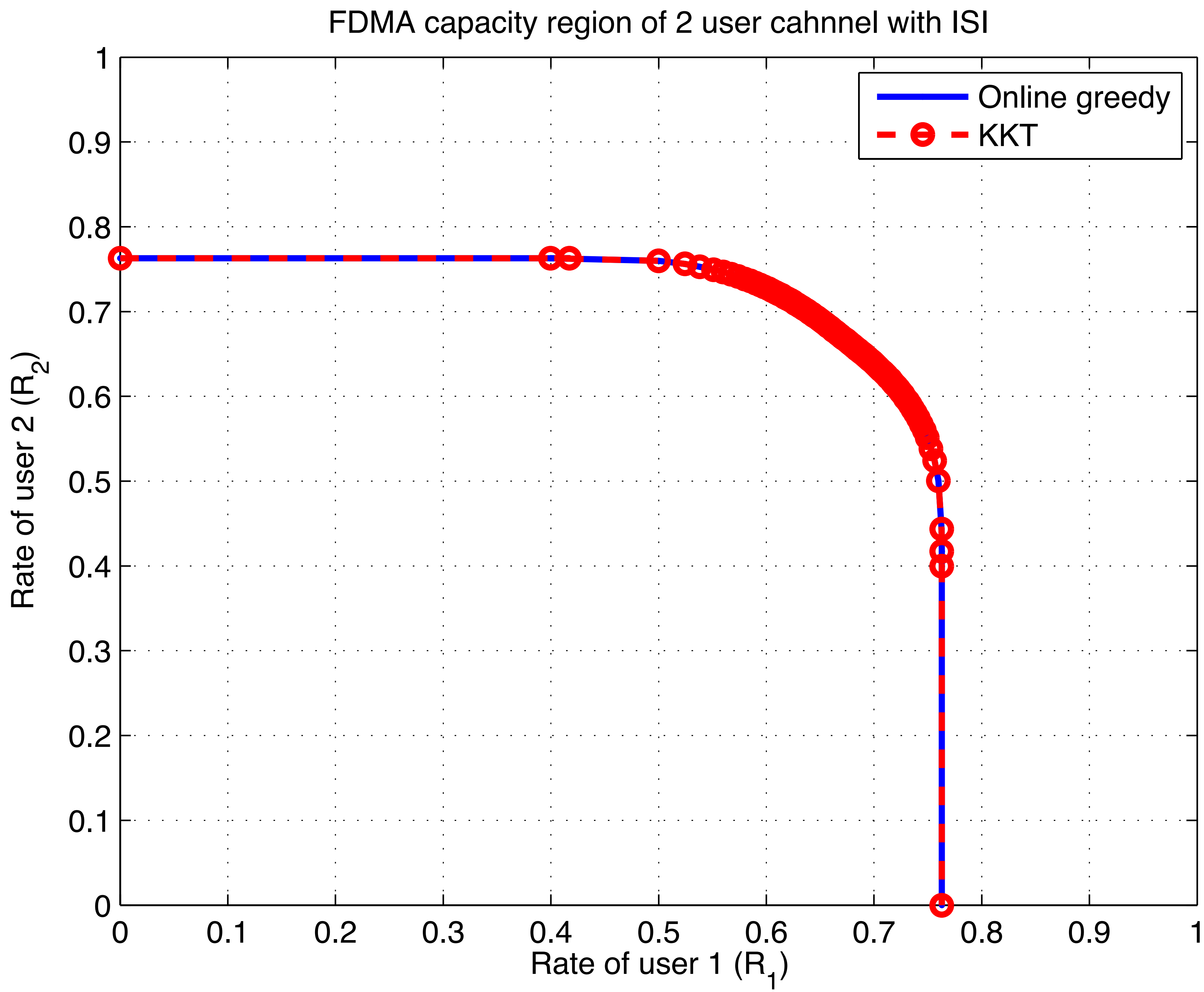
a majorizes **b** if $\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} b_i, \forall\ k \leq n$

Karamata's inequality $\sum_{i=1}^{n} g(a_i) \geq \sum_{i=1}^{n} g(b_i)$ for any *convex* function g

Proof very specific to *log* utility

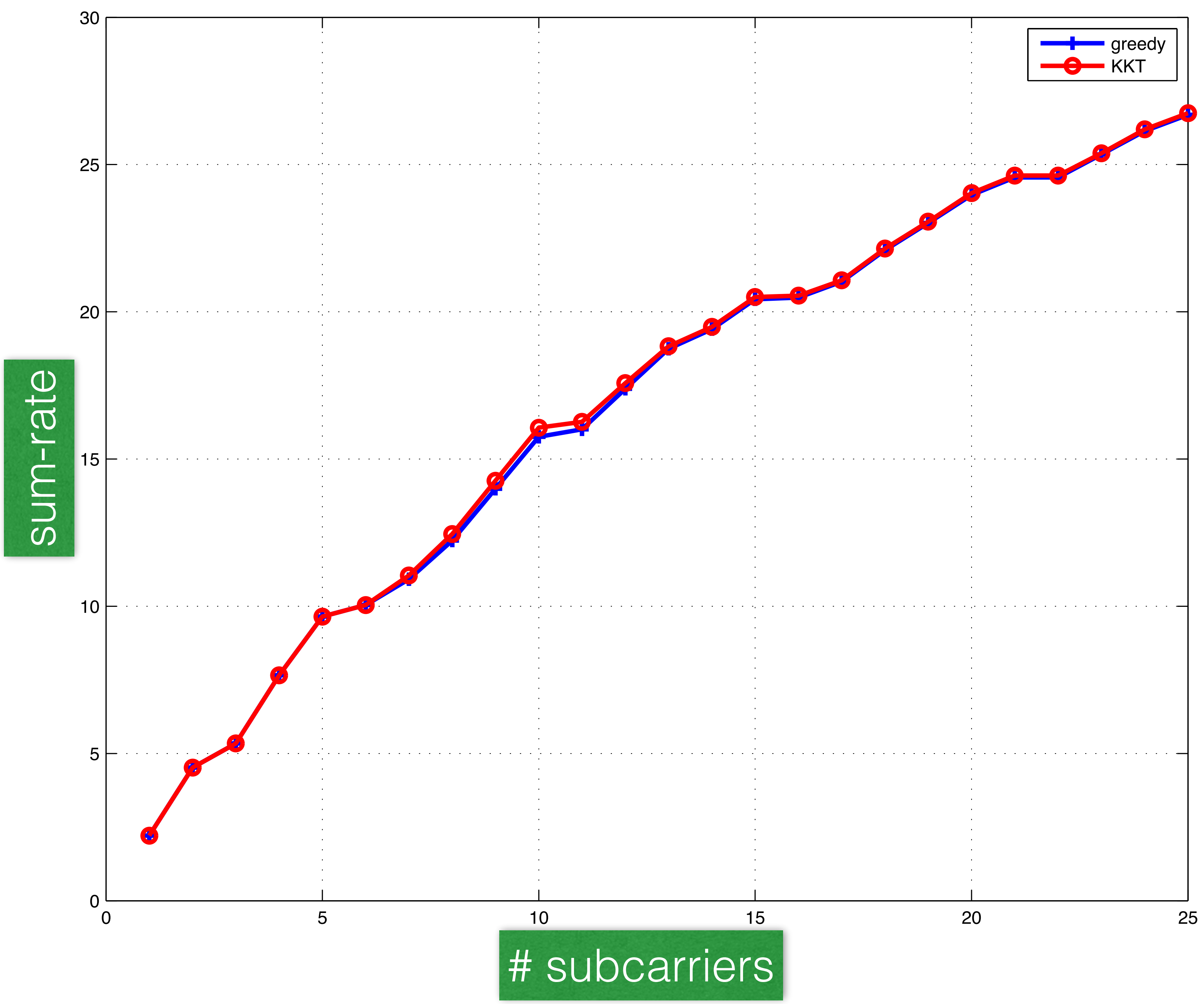
Simulations: Capacity

Comparison with Heuristics



Simulations: Sum-rate

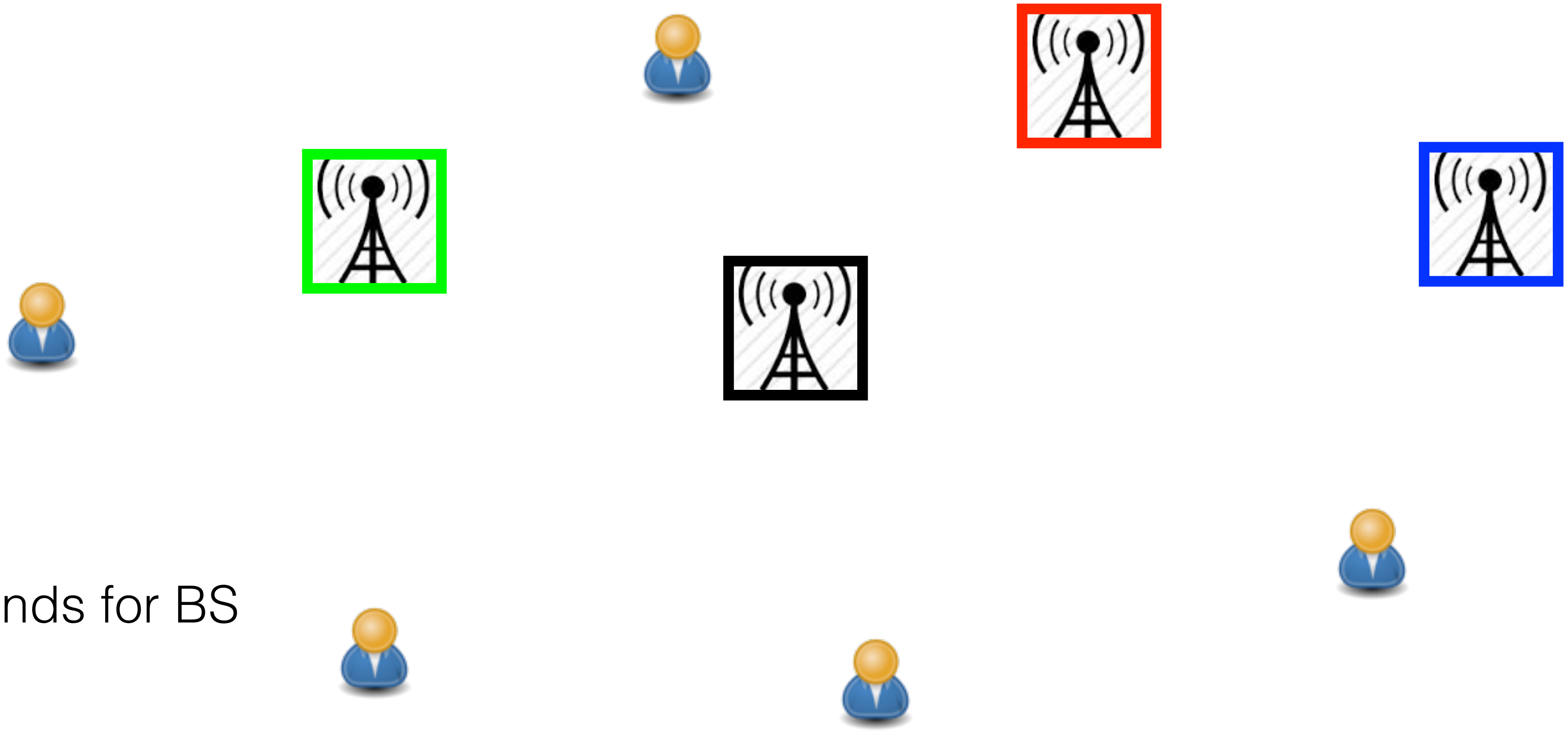
Comparison with Heuristics with 10 users



What more !

Downlink BS association

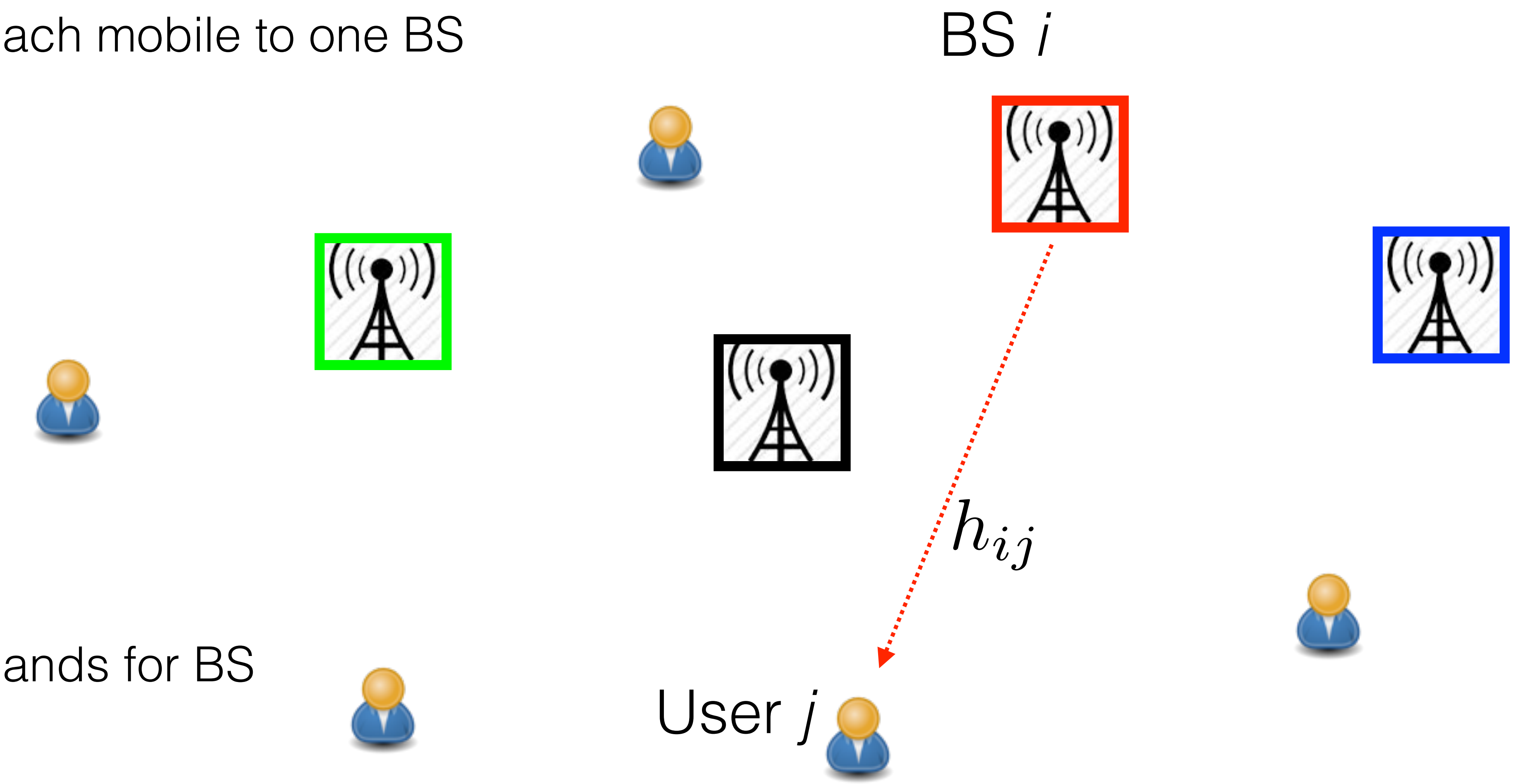
Associate each mobile to one BS



Disjoint bands for BS

Downlink BS association

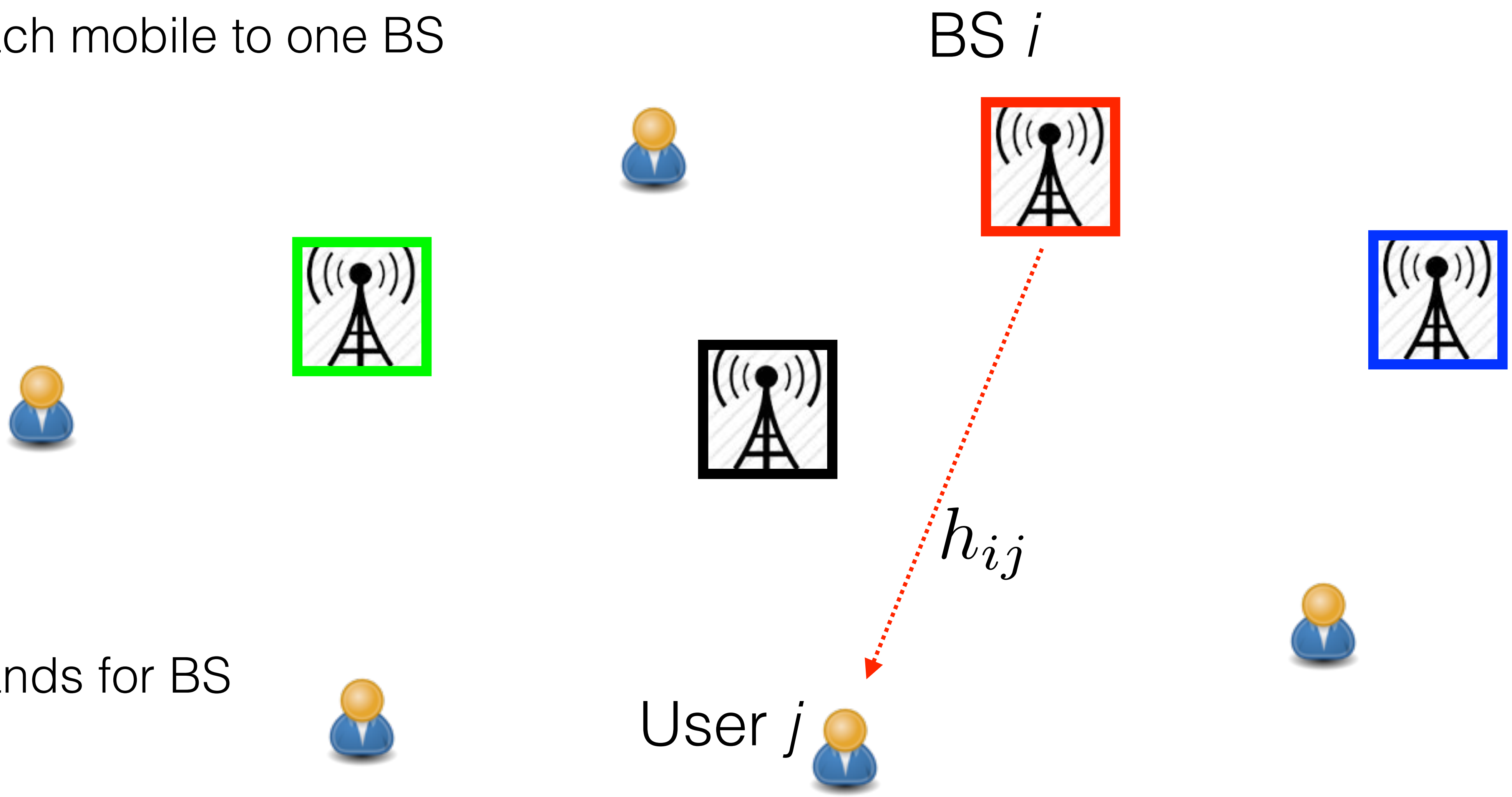
Associate each mobile to one BS



Disjoint bands for BS

Downlink BS association

Associate each mobile to one BS



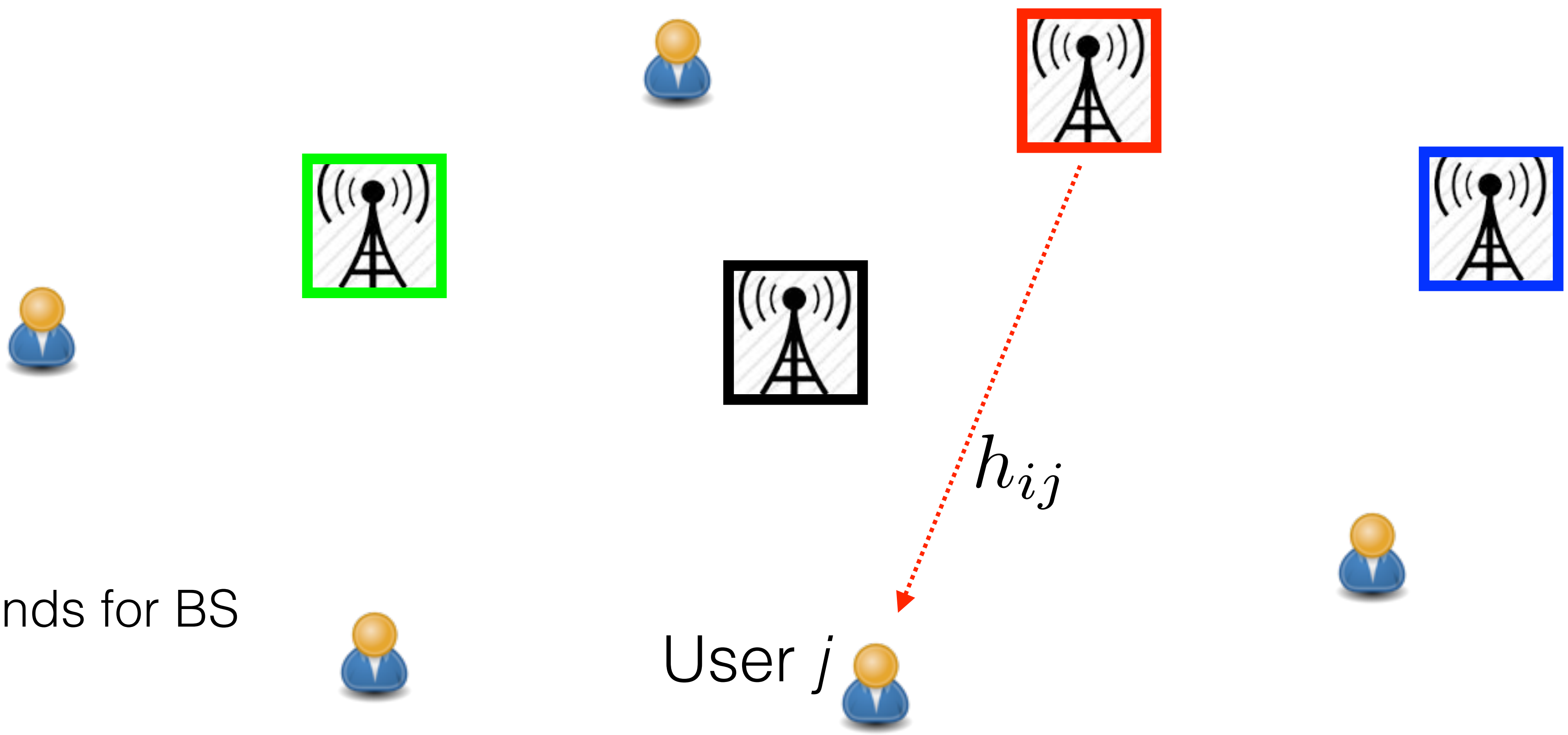
Disjoint bands for BS

Each BS gives power according to log utility for all associated users

$$R_i = \max_{\sum_{j \in A_i} P_j \leq P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Downlink BS association

Associate each mobile to one BS



Disjoint bands for BS

Each BS gives power according to log utility for all associated users

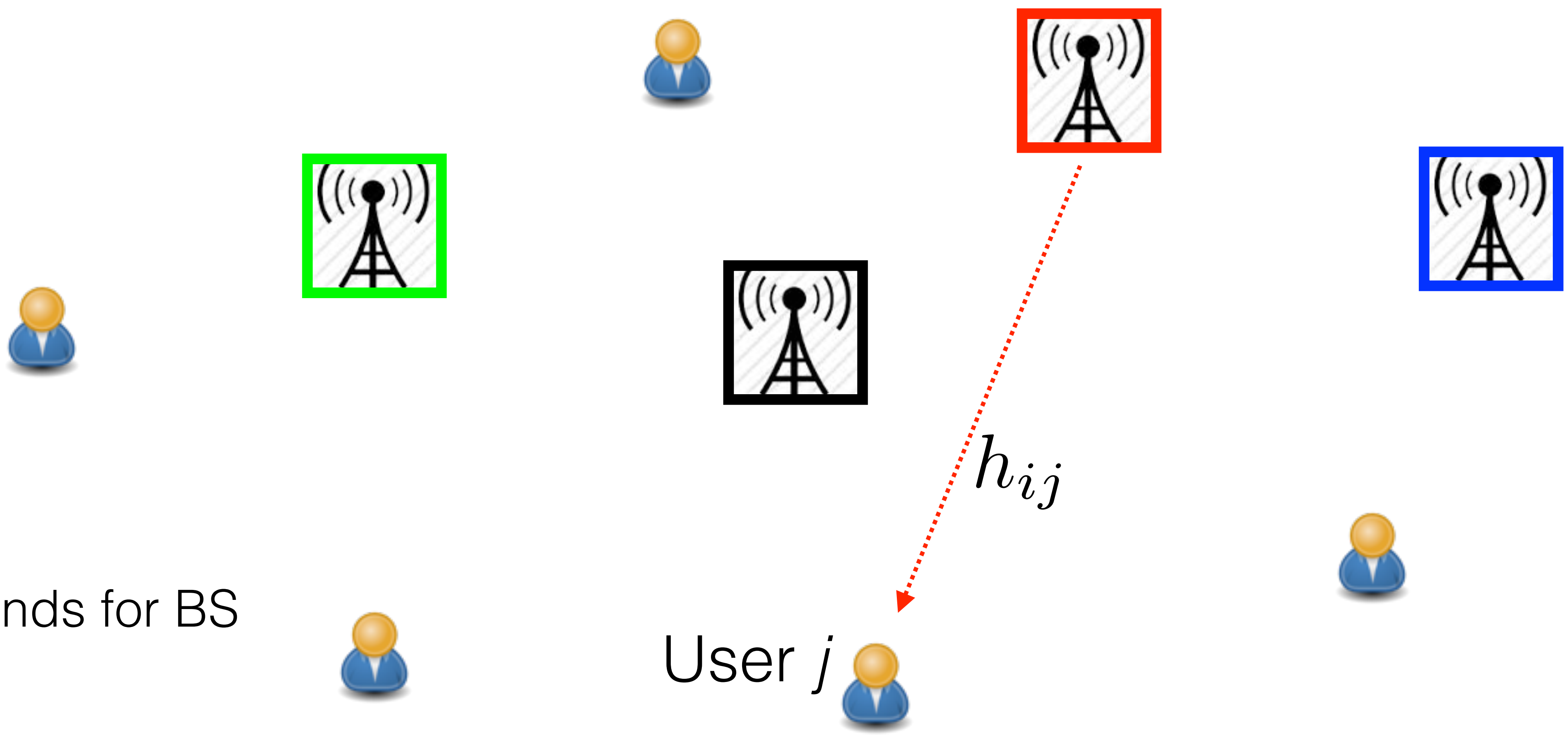
$$R_i = \max_{\sum_{j \in A_i} P_j \leq P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Find association to maximize the sum-rate

$$\max_{A_i} \sum_{i=1}^{m} R_i$$

Downlink BS association

Associate each mobile to one BS



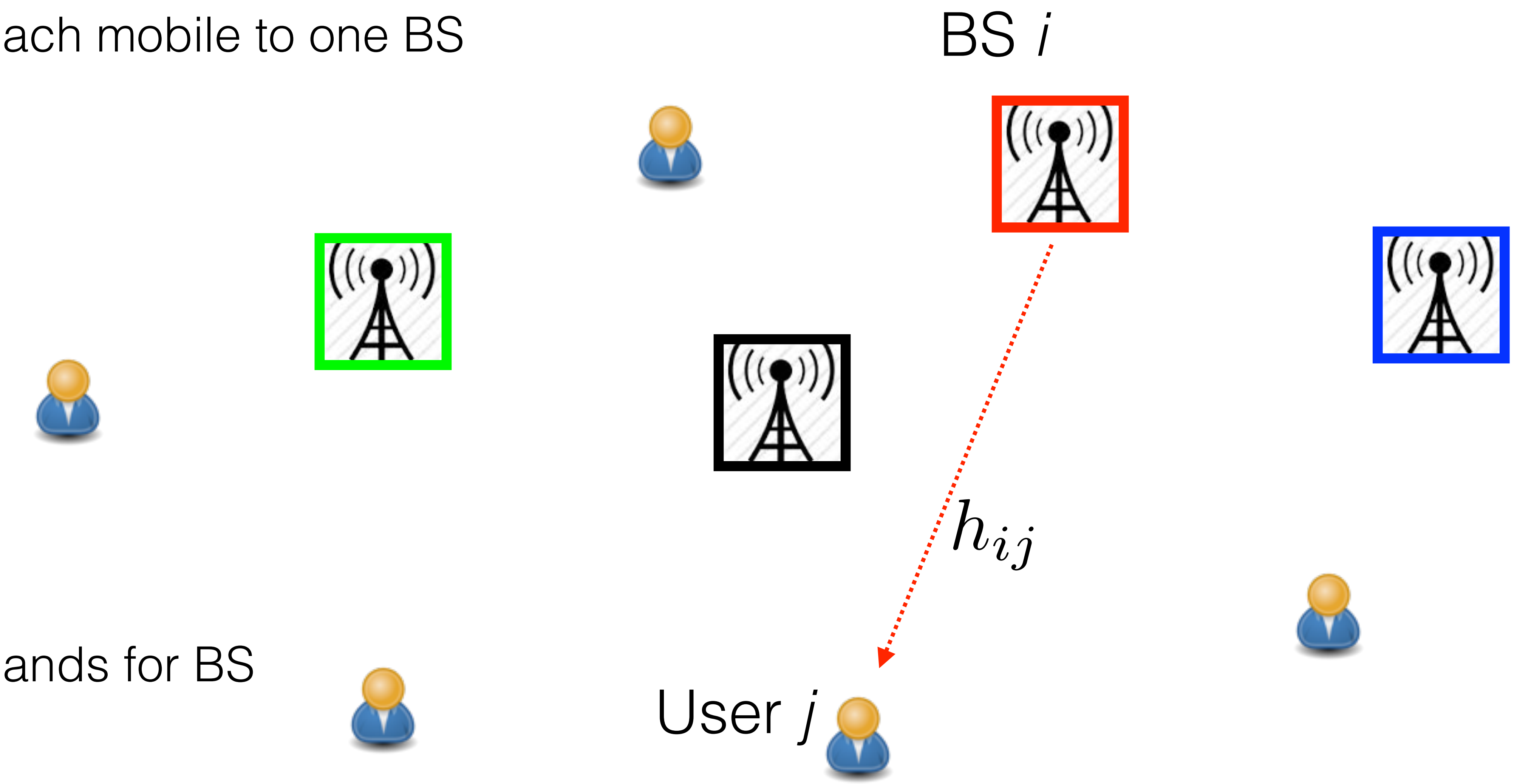
Disjoint bands for BS

Each BS gives power according to log utility for all associated users

$$R_i = \max_{\sum_{j \in A_i} P_j \le P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Downlink BS association

Associate each mobile to one BS



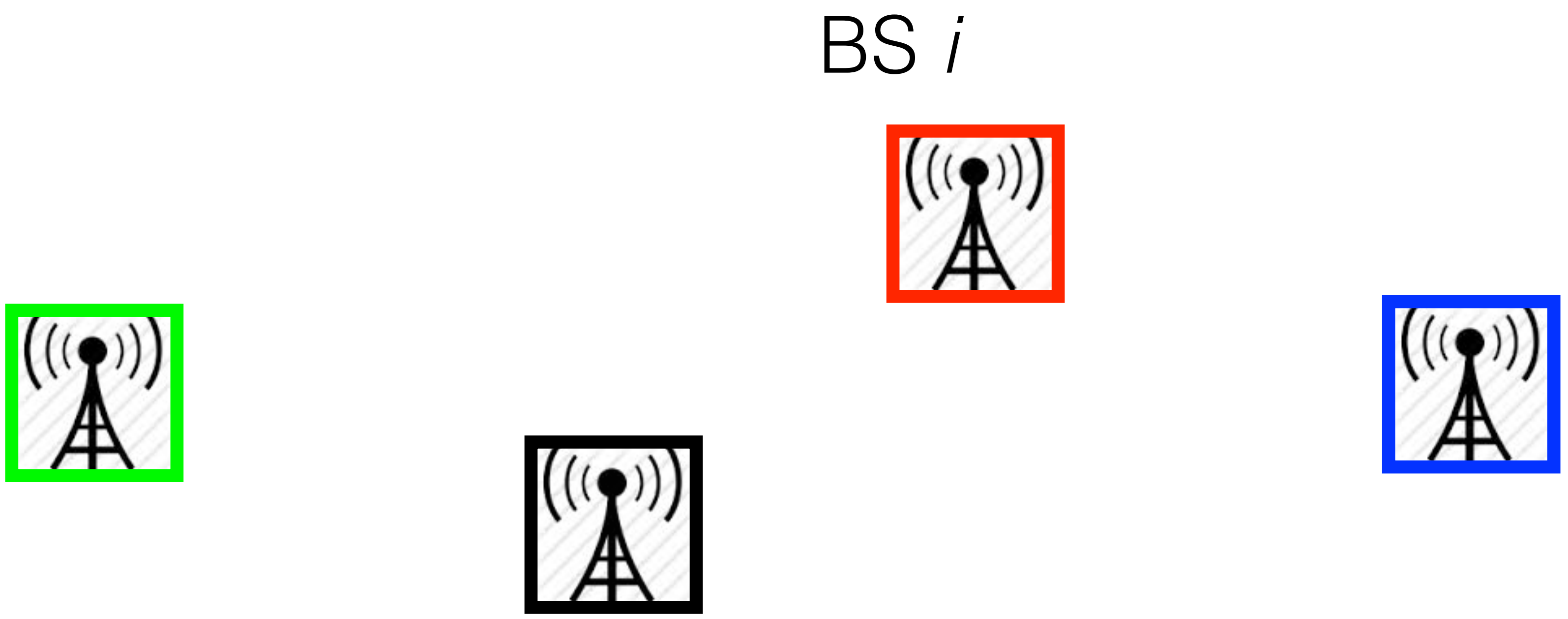
Disjoint bands for BS

Each BS gives power according to log utility for all associated users

$$R_i = \max_{\sum_{j \in A_i} P_j \leq P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Again 1/2 Approx.

Online Downlink BS association

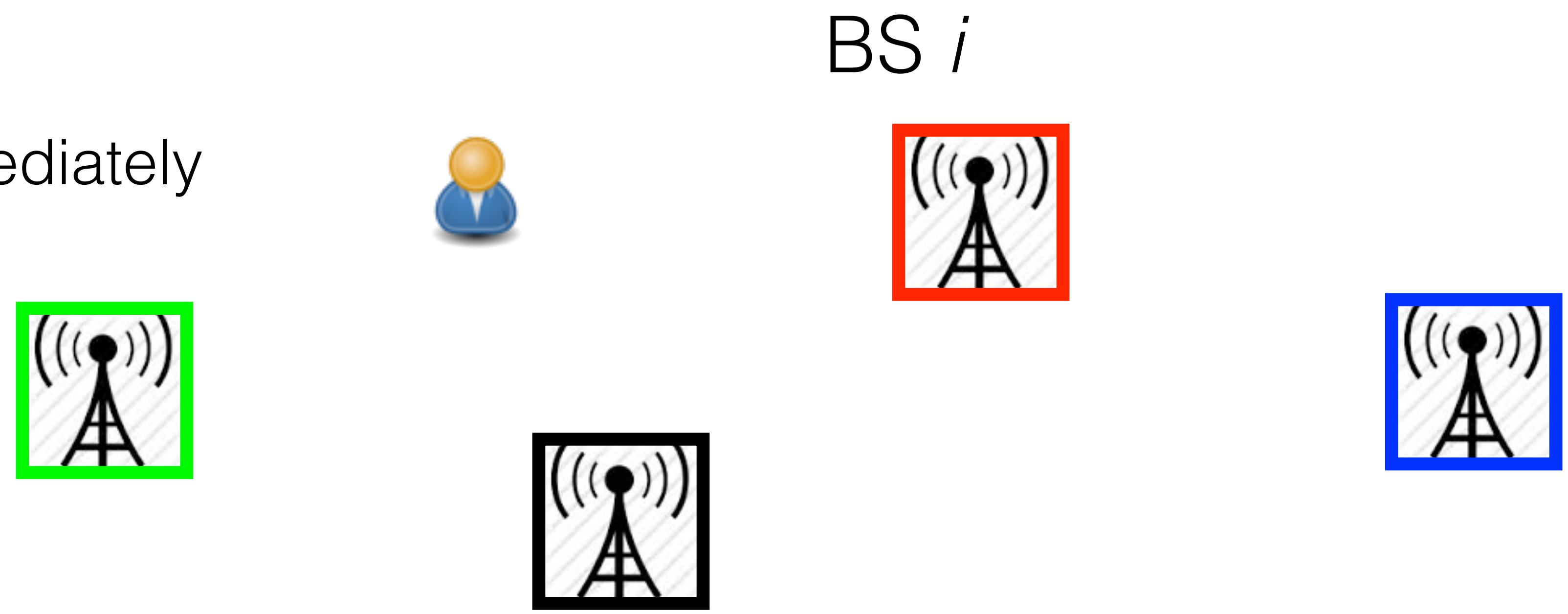


Disjoint bands for BS

Online Downlink BS association

users appear one by one

associate to one BS immediately



Disjoint bands for BS

Online Downlink BS association

users appear one by one

associate to one BS immediately

BS i

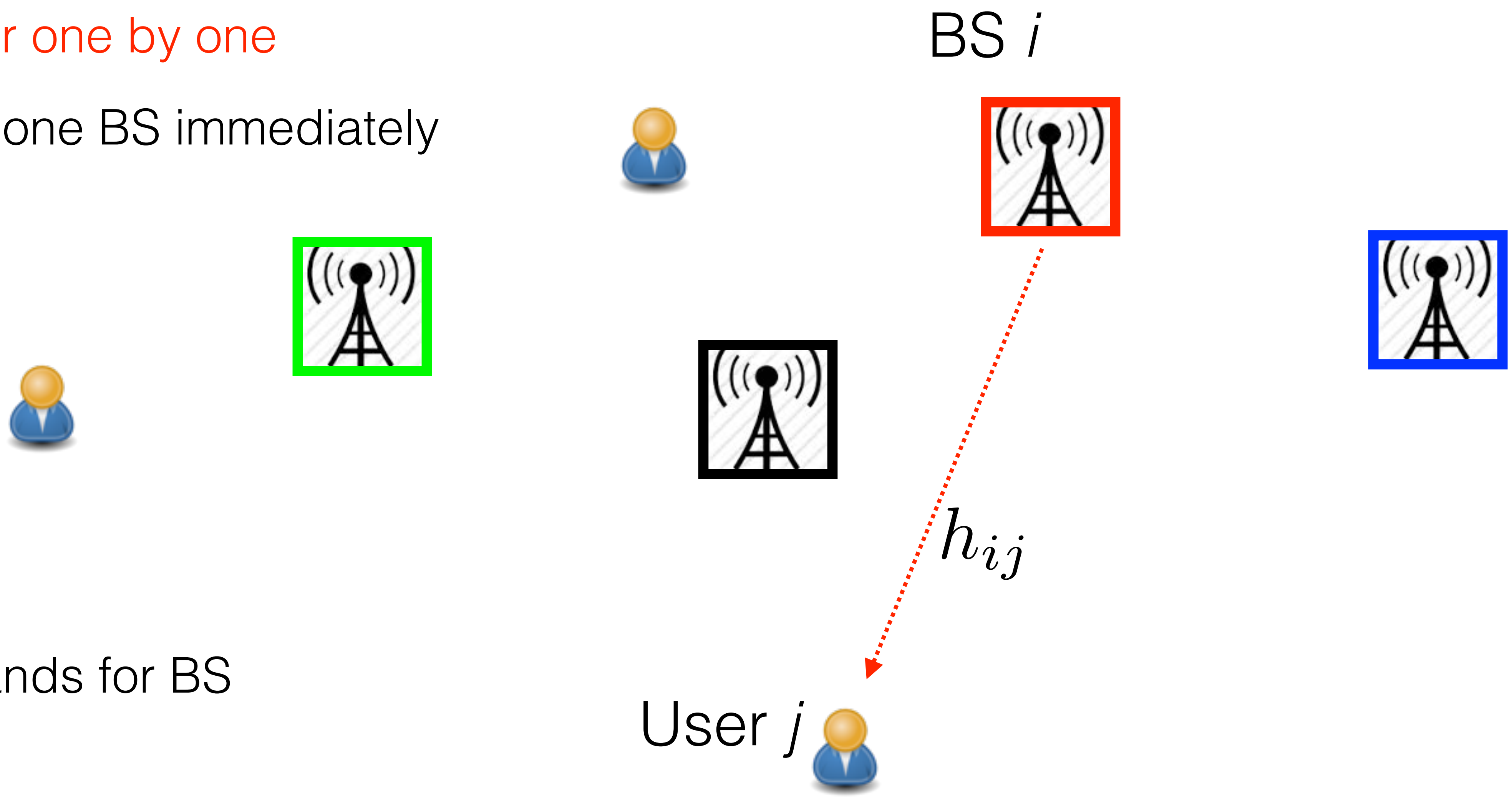
Disjoint bands for BS

Online Downlink BS association

users appear one by one

associate to one BS immediately

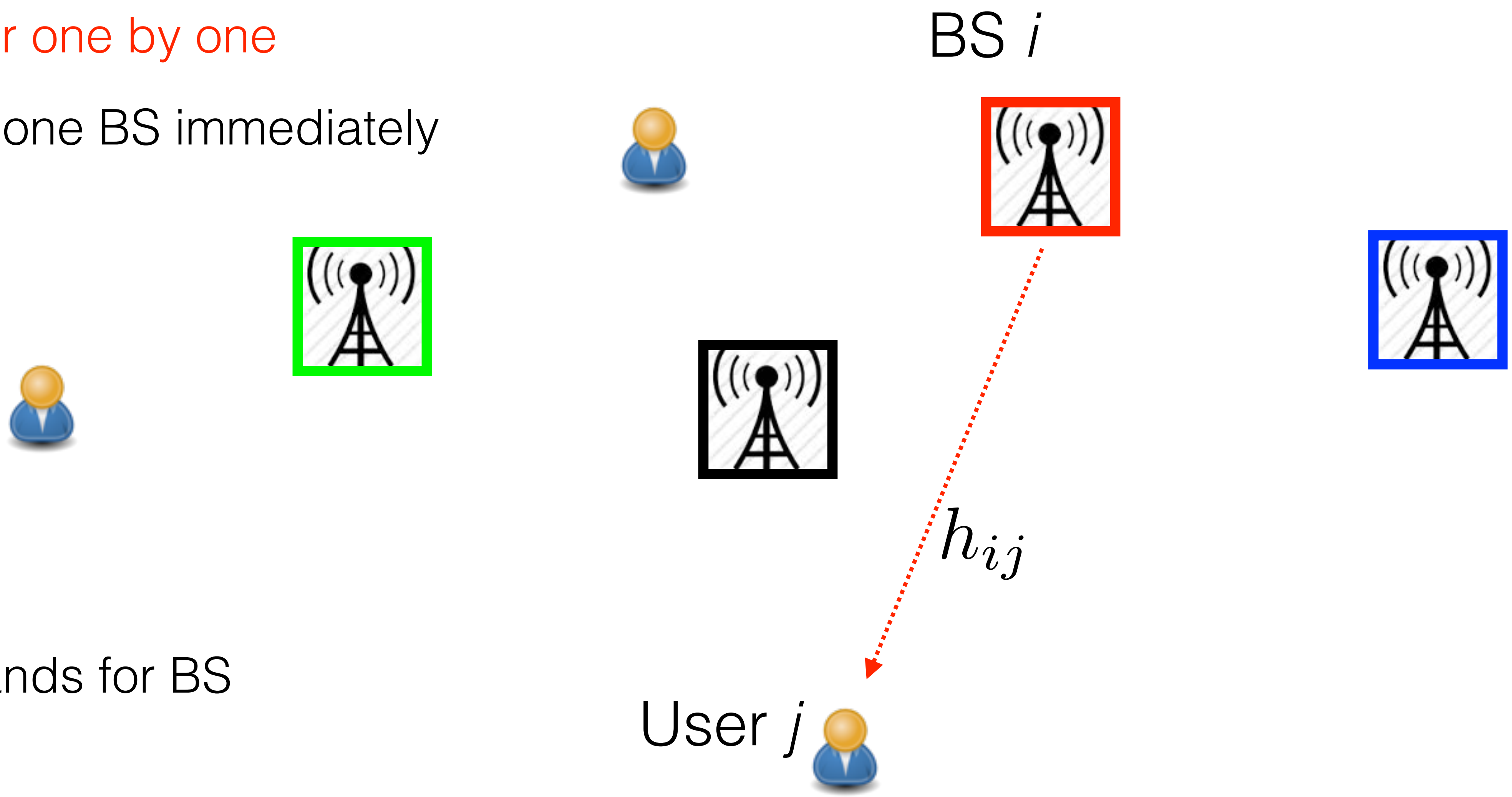
Disjoint bands for BS



Online Downlink BS association

users appear one by one

associate to one BS immediately



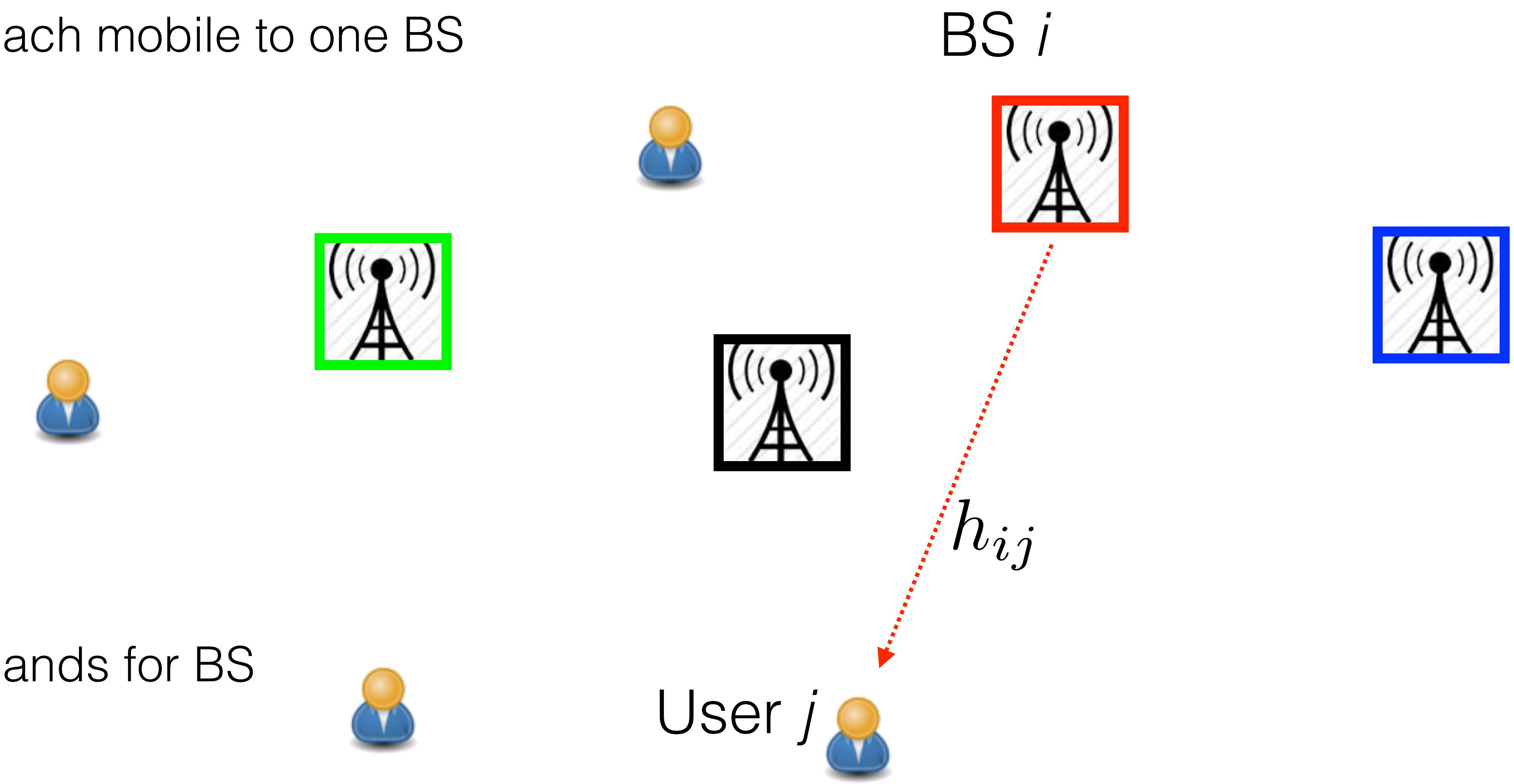
Disjoint bands for BS

Since the greedy algorithm works with one element (user) at a time

Again 1/2 Approx.

Untruthful Users Downlink BS association

Associate each mobile to one BS



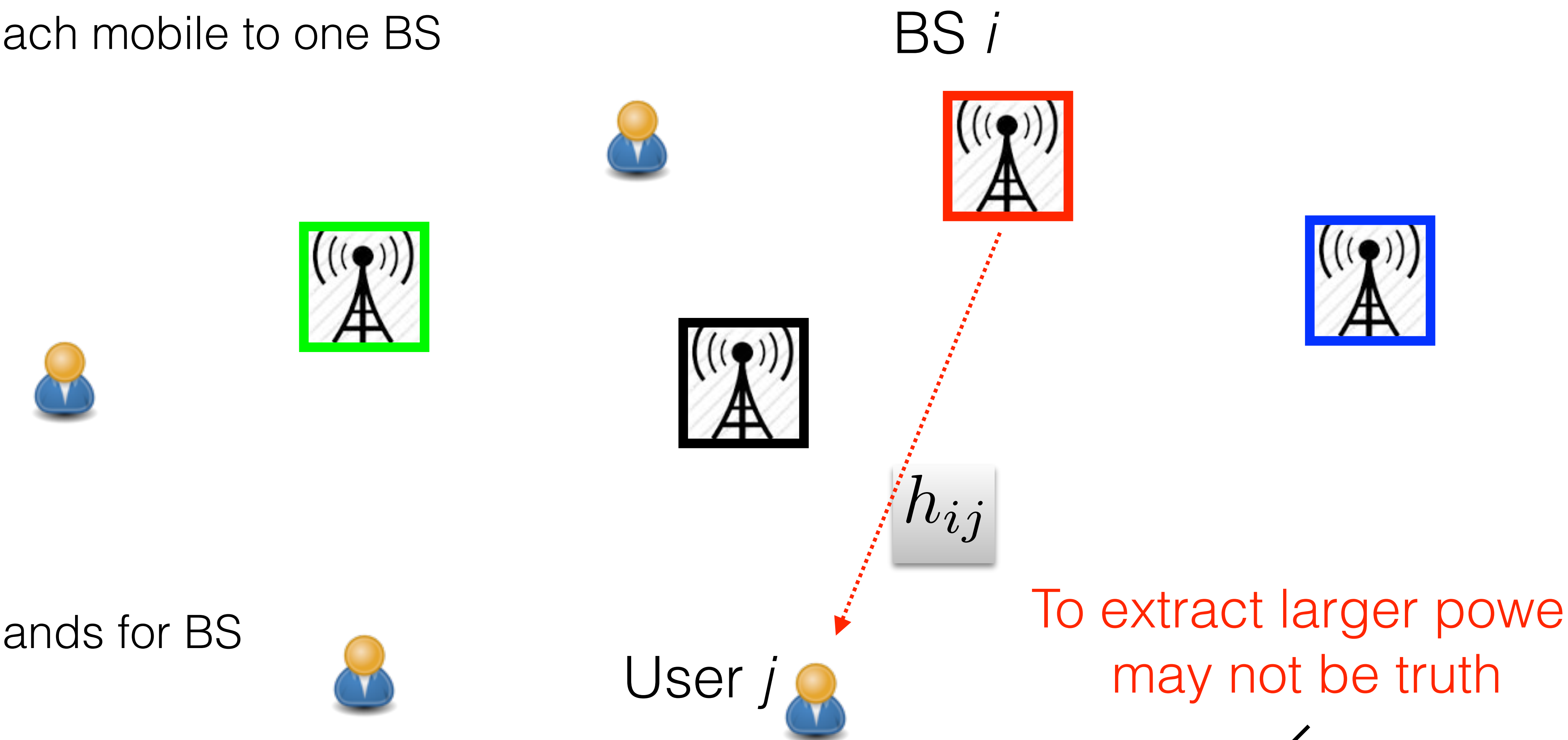
Disjoint bands for BS

Each BS gives power according to log utility for all associated users

$$R_i = \max_{\sum_{j \in A_i} P_j \leq P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Untruthful Users Downlink BS association

Associate each mobile to one BS



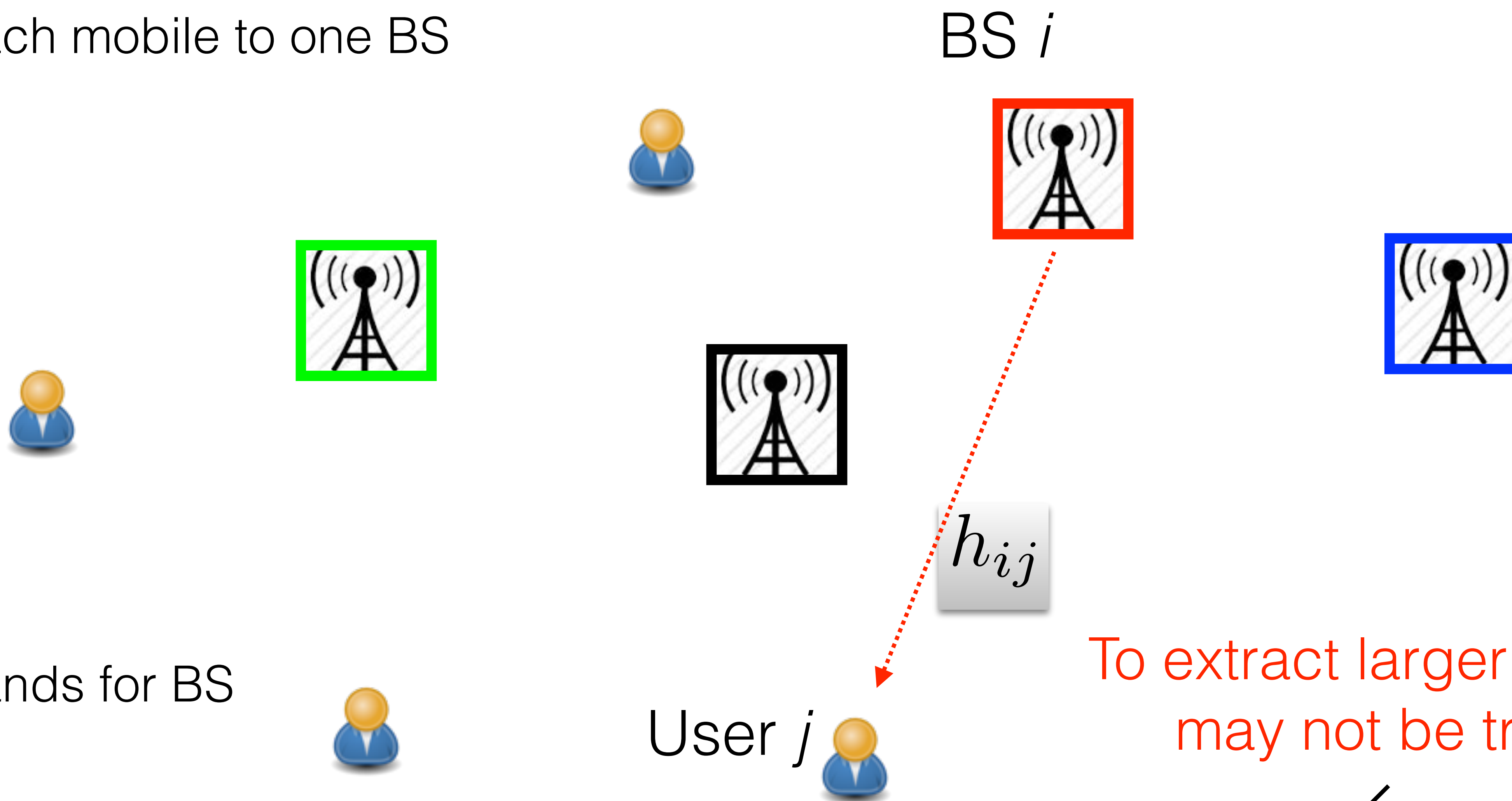
Disjoint bands for BS

Each BS gives power according to log utility for all associated users

$$R_i = \max_{\sum_{j \in A_i} P_j \leq P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Untruthful Users Downlink BS association

Associate each mobile to one BS



Disjoint bands for BS

To extract larger power may not be truth

Each BS gives power according to log utility for all associated users

$$R_i = \max_{\sum_{j \in A_i} P_j \leq P} \sum_{j \in A_i} \log(1 + P_j h_{ij})$$

Using VCG pricing again 1/2 Approximation

Lot more !