UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

LOCALIZZAZIONE DI SMARTPHONE MEDIANTE SEGNALI DI CONTROLLO 5G

Relatore:
Prof. Stefano Tomasin

Laureando:
Leonardo Fasolo

ANNO ACCADEMICO: 2022/2023

Data di laurea: 12-03-2024

Indice

1 Determinazione della posizione di un utente **3**

1.1 Il mezzo trasmissione 4

1.2 Strategia di localizzazione dell'utente 8

2 Ricostruzione della mappa ed algoritmi proposti **11**

2.1 L'algoritmo 1 13

2.1.1 Versione semplificata 15

2.1.2 Versione con un cluster 17

2.2 Algoritmo 2 19

2.2.1 Risoluzione delle precedenti problematiche 19

3 Prestazioni degli algoritmi **27**

3.1 Casi particolari 27

3.1.1 Un cluster sviluppato su più righe e colonne 27

3.1.2 Due o più cluster 29

3.2 Tabella delle prestazioni 31

4 Conclusioni **32**

Sommario

Nei sistemi 5G è stata posta molta enfasi alla localizzazione, che però pone problemi di privacy se anche altri dispositivi possono facilmente localizzare gli utenti. In questa tesi si considera la localizzazione di un utente tramite l'ascolto di segnali di feedback da lui trasmessi alla stazione base.
Per poter fare ciò è necessario prima costruire, attraverso i segnali radio trasmessi dagli utenti tramite i loro dispositivi, una mappa che descrive la copertura di una stazione base. La tesi propone una tecnica per ricostruire la mappa a partire dalla raccolta dei segnali muovendo l'utente a caso nello spazio. L'algoritmo verrà analizzato in diversi contesti, delineando i casi in cui non è possibile ricostruire la mappa o casi in cui sono presenti degli errori nella ricostruzione.

Introduzione

Si cerca di simulare un tentativo di localizzazione di uno specifico utente tramite l'*ascolto* di segnali di feedback trasmessi dall'utente ad una stazione base. Per poter fare ciò è necessario prima costruire, attraverso i segnali radio trasmessi dagli utenti tramite i loro dispositivi, una mappa che descrive la copertura di una stazione base. Ciò che si conosce sono una sequenza di guadagni di canale tra un dispositivo e la stazione base, ad una determinata frequenza. Ogni valore, per maggiore chiarezza, viene associato ad un determinato colore all'interno della mappa, ma non si sa la sua posizione precisa nella mappa. Questa sequenza ha una lunghezza che varia da utente ad utente e nelle varie realizzazioni. L'utente può muoversi all'interno di tutta la mappa, tornando anche sui suoi passi o ripetere lo stesso percorso. Si ipotizza che l'utente abbia velocità costante. Si hanno a disposizione infinite sequenze di utenti per ricostruire la mappa. Nella prima situazione sappiamo che sono presenti celle tutte di valori diversi ad eccezione di un cluster di celle, tutte adiacenti tra di loro, all'interno della mappa che hanno lo stesso valore. L'obiettivo è quello di conoscere la forma del cluster di celle e ricostruire il resto della mappa.

Capitolo 1

Determinazione della posizione di un utente

Nei sistemi 5G è stata posta molta enfasi sulla localizzazione degli utenti. Ad esempio, per localizzare un utente si può sfruttare il fatto che la rete cellulare riceve i segnali radio trasmessi dall'utente attraverso più stazioni base, alle volte dotate di molteplici antenne. Mediante tecniche di triangolazione/-trilaterazione è possibile localizzare l'utente con buona precisione. Inoltre, il passaggio dalle microonde nelle reti di quarta generazione (4G) alle mm-Waves nelle reti 5G, e successivamente alle onde submillimetriche nelle reti di sesta generazione (6G), apre la strada ad una precisione di localizzazione nell'ordine dei centimetri. Purtroppo questa novità ha interessato anche dei potenziali utenti malevoli, che in seguito chiamaremo attaccanti, disposti a carpire la posizione dell'user equipment (UE) violando così la privacy degli utenti.

1.1 Il mezzo trasmissione

Si utilizzi la figura 1 come riferimento, all'interno della quale sono presenti la *base station*, punto di connessione attraverso il quale i dispositivi elettronici possono comunicare tra loro, posto al centro del cerchio; e l'apparecchiatura di un utente vittima dell'attacco (user equipment, UE).
La base statica di nuova generazione (new generation node base, gNB) si trova al centro del cerchio, che coincide con l'origine delle coordinate $(0, 0)$, mentre la posizione dell'UE ha coordinate $p = (px, py)$. Il gNB è dotato di un array lineare di N antenne a polarizzazione incrociata ($2N$ antenne in totale), mentre l'UE è dotato di un array lineare di N antenne. Secondo la versione 16 dello standard di rete cellulare 3GPP, la modulazione a divisione di frequenza ortogonale (OFDM) viene utilizzata nel downlink e le sottoportanti sono raggruppate in sottobande.

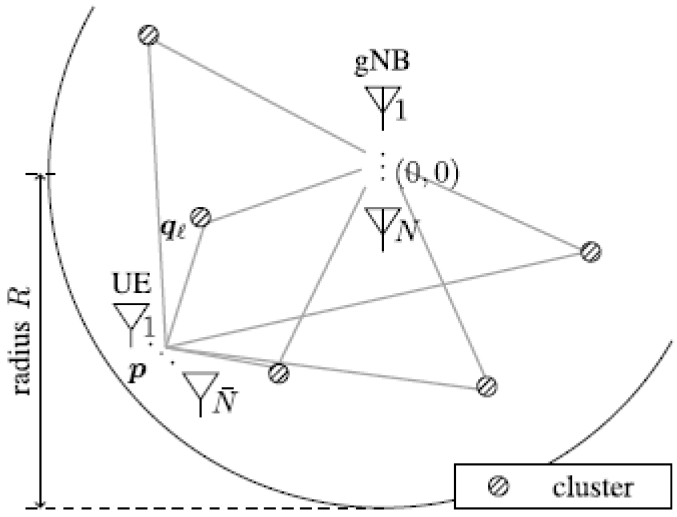

Figura 1: Descrizione schematica del modello di canale per una cella circolare di raggio R, e un generico cluster di coordinate q.

La tipologia di attacco che si andrà a trattare in questa tesi è quella dell'intercettazione di un segnale demodulato proveniente dall'utente e dalle

stazioni base con cui l'utente stesso comunica. Questa tipologia di attacco non richiede l'utilizzo di più antenne o hardware di alto livello. Inoltre l'implementazione richiede solo la demodulazione e la decodifica del segnale digitale. È proprio per questa sua bassa comnplessità che rende accessibile questo tipo di attacco ad un numero maggiore di utenti malevoli. Tuttavia, gli attacchi svolti semplicemente ricevendo i segnali trasmessi dall'utente forniscono una posizione approssimativa della cella su cui si trova l'utente.
Nell'articolo *Localization attack by precoder feedback overhearing in 5G networks and countermeasures* [1] viene considerata nel dettaglio una variante di questa strategia. Viene proposto un nuovo attacco di localizzazione, basato sull'intercettazione del segnale di controllo trasmesso dagli utenti e relativo alle informazioni sullo stato del loro canale, come previsto dalle specifiche dello standard NewRadio (NR) 3GPP delle reti 5G.
In particolare, il 3GPP fornisce all'utente i mezzi per stimare il canale di downlink (dalla gNB all'UE) e quindi permette di scegliere il miglior precoder (da utilizzare alla gNB per la trasmissione dei segnali) da un codebook predefinito; infine, l'utente restituisce alla stazione base l'indice del precoder selezionato. Il precoder, infatti, è legato al canale sperimentato dall'UE, e quindi rivela, in parte, la sua posizione.
Il segnale che l'UE riceve può essere espresso in questo modo:

$$\boldsymbol{r}_k(k) = \boldsymbol{H}(\boldsymbol{p}, k)\boldsymbol{w}(k)s_k(k) + \boldsymbol{v}_k(k), \quad k = 1, \ldots, K \tag{1.1}$$

dove $\mathbf{r}_k(k)$ rappresenta il segnale ricevuto dall'utente sulla sottobanda k, $\mathbf{H}(\boldsymbol{p}, k)$ rappresenta la matrice di canale tra l'utente e la stazione base di servizio sulla sottobanda k nella posizione $\boldsymbol{p}$, $\mathbf{w}(k)$ rappresenta il vettore di precodifica sulla sottobanda k, $s_k(k)$ rappresenta il simbolo trasmesso sulla sottobanda k e $\mathbf{v}_k(k)$ rappresenta il rumore additivo gaussiano bianco

(AWGN) sulla sottobanda k.

Analisi della metodologia di localizzazione:

Nella versione 16 dello standard 3GPP sono definiti vari tipi di codebook. L'unico tipo di codebook che le UE saranno obbligate a rispettare è il codebook di tipo I. Si andrà ad analizzare solamente il funzionamento di quest'ultimo poiché un utente malevolo andrà a sfruttare questo vincolo. Ciascun vettore di precodifica è identificato dalla coppia *(m, n)*. L'indice m identifica il seguente vettore vettore

$$\tilde{\boldsymbol{w}}_m = \left(1, \mathrm{e}^{\left(\frac{j2\pi m}{NO}\right)}, \ldots, \mathrm{e}^{\left(\frac{j2\pi(N-1)m}{NO}\right)}\right)^T. \tag{1.2}$$

L'indice n è invece associato all'angolo di cofasamento tra le polarizzazioni delle coppie di antenne $\psi_n = \exp(j2\pi n/4)$. Successivamente, il vettore di precodifica associato alla coppia (m, n) è definito come:

$$\boldsymbol{w}_{m,n}c = \frac{1}{\sqrt{N}} \begin{pmatrix} \tilde{\boldsymbol{w}}_m \\ \\ \psi_m \tilde{\boldsymbol{w}}_m \end{pmatrix}. \tag{1.3}$$

La costruzione delle tuple *(m, n)* varia ta le differenti modalità di *feedback*.

- **Modalità di *feedback* 1:**

  Nella modalità di feedback 1, viene utilizzato lo stesso vettore di precodifica per tutte le sottobande, mentre viene applicata una cofasatura dipendente dalla sottobanda delle due polarizzazioni. L'UE seleziona il precodificatore come soluzione del seguente problema di massimizzazione della velocità di trasmissione:

$$(w^*, \{n^*(k)\}) = \arg\max R_{m,n(k)} \left[\left\{w(k) = w_{m,n(k)}\right\}\right], \tag{1.4}$$

dove $R_{m,n(k)}$ rappresenta l'efficenza spettrale di ogni sottoportante della sottobanda k e dove $m \in M$ e $n(k) \in N$, per ogni sottobanda k. Il feedback del gNB comprende due indici, che per la prima modalità sono definiti come segue:

- $i_{1,1}$ è la rappresentazione intera senza segno $log_2(NO)$-bit di $m^* \in M$.
- $i_2(k), k = 1, ..., K$, è un pacchetto a 2 bit $n^*(k)$. Questo richiede in tutto 2000 bit.

- **Modalità di *feedback* 2:**

  La scelta del vettore di precodifica e cofasatura dipende dalla sottobanda. La cofasatura viene scelta come per la modalità precedente, e l'UE sceglie il precodificatore tramite il problema di massimizzazione del troughput:

$$(m^*, \{\delta^*(k)\}, \{n^*(k)\}) = \text{argmax}_{m,\{\delta(k)\},\{n(k)\}} R\times\left[\left\{\boldsymbol{w}(k) = \boldsymbol{w}_{2m+\delta(k),n(k)}\right\}\right], \tag{1.5}$$

  $m \in \{0, \ldots, (NO)/2 - 1\}$ e $\delta(k) \in \{0, ..., 3\}$ Per quanto riguarda il feedback del gNB abbiamo:

  - $i_{1,1}$ è la rappresentazione intera senza segno $log_2(NO)$-bit di $m^* \in$ M.
  - $i_2(k), k = 1, ..., K$ è un insieme di 4 bit, di cui 2 più significativi e due meno, che trasportano il valore intero e senza segno di δ^*(k)

e n^*(k). Questo richiede un totale di 4000 bit, il doppio del caso precedente.

- **Modalità di *feedback* 3:**
  Nella terza modalità di feedback, si assegna ad ogni sottobanda un vettore di precodifica differente, quindi con entrambi i valori di $m(k)$ e $n(k)$ differenti. La coppia di indici per ogni sottobanda viene trovata grazie a:

$$(m^*(k), \{n^*(k)\}) = \text{argmax}_{m(k),\{n(k)\}} R \times \left[\left\{\boldsymbol{w}(k) = \boldsymbol{w}_{m(k),n(k)}\right\}\right], \quad (1.6)$$

  Per quanto riguarda il feedback del gNB abbiamo:

  - $i_2(k), k = 1, ..., K$ è un intero senza segno, del quale $\log_2$(NO)bit rappresentano $m(k)$, e i due bit meno significativi $n(k)$. Ciò richiede un numero complessivo di $K(log_2(NO) + 2)$ bit.

1.2 Strategia di localizzazione dell'utente

L'attaccante è dotato di un dispositivo in grado di captare lo scambio di segnali tra il gNB e l'UE. Si assuma che l'attaccante intercetti il feedback del precodificatore della UE, quindi $\{i_{1,1}\}$ e $\{i_2(k)\}$. Gli UE nella stessa posizione, sperimentano polarizzazioni e rumore differenti, producendo valori di $\{i_{1,1}\}$ e $\{i_2(k)\}$ diversi ma comunque appartenenti ad un sottoinsieme ristretto di valori. Dato che l'attaccante è interessato solo alla posizione della UE, trascurerà le informazioni riguardanti la polarizzazione, ovvero $\{i_2(k)\}$ nella modalità di feedback 1 e nelle modalità 2 e 3 i due bit meno significativi

di $\{i_2(k)\}$.

L'obiettivo dell'attaccante è quello di intercettare il vettore di feedback b^* inviato dalla UE da una posizione sconosciuta. Per poterlo fare, l'attaccante trova la posizione dell'UE calcolando la probabilità di ottenere in feedback il versore β dato che l'UE è in possesso di $\underline{\mathbf{s}}$, cioè:

$$p_{map}(\beta, \boldsymbol{s}) = P[b(s) = \beta] \tag{1.7}$$

Per poter raccogliere questi vettori di feedback β è necessario raccogliere tutti i bit di feedback del precodificatore all'interno della cella in cui si trova l'UE.

Sia S l'insieme delle posizioni esplorate durante questa fase. Si definisca la probabilità di occorrenza del vettore β come:

$$p(\beta) = E[P[b(\boldsymbol{p}) = \beta]] \tag{1.8}$$

Questa probabilità fornisce una mappa probabilistica dei vettori di feedback alle posizioni nella cella e sarà sfruttata dall'attaccante per localizzare l'UE vittima nella fase successiva. Viene definita anche la probabilità di occorrenza del vettore β come:

$$\hat{p}(\beta) = (\hat{p}_x, \hat{p}_y) = \sum_{s \in S} \frac{p_{map}(\beta, s)}{p(\beta)} \boldsymbol{s} \tag{1.9}$$

dove il valore atteso è assunto rispetto alla posizione $\boldsymbol{p}$.

Dopo aver raccolto tutti i vettori è possibile, tramite un algoritmo che verrà spiegato nel dettaglio in seguito, ricostruire interamente la mappa probabilistica. L'attaccante attraverso questa mappa potrà, come già detto in precedenza, intercettare il vettore di feedback dell'UE, verificare in quale zona della mappa si trova e quindi conoscerne la posizione.

Capitolo 2

Ricostruzione della mappa ed algoritmi proposti

Per il funzionamento della rete 5G i dispositivi si collegano a una stazione base, e il canale di comunicazione è quantizzato mediante un codebook. Questa versione quantizzata è trasmessa dal dispositivo alla stazione base e viene trasmesso in chiaro.
Un attaccante è in grado di ascoltare passivamente questa comunicazione tra il dispositivo dell'utente e la stazione base ed intercettare le informazioni non crittografate. Nello specifico ciò che viene intercettato è il valore quantizzato di feedback di un precodificatore (in seguito definito precoder) associato all'utente in questione, spiegato nel paragrafo 1.2. Si suppone che il terminale utilizzato per raccogliere i valori di feedback si muova a velocità costante, ma in posizioni ignote nella rete. Inoltre, le sequenze di valori quantizzati raccolte sono parziali, ovvero non coprono tutta la mappa, e discontinue, cioè sono raccolte in generale in posizioni diverse della mappa.
Innanzitutto i segnali intercettati vengono utilizzati per creare una matrice di adiacenza, ovvero una matrice in cui l'elemento *(i,j)* generico vale 1 se il

valore quantizzato di i è adiacente (nel movimento di un utente) al valore quantizzato j. Quest'ultima poi viene usata per ricreare una mappa di indici di precoder ricevuti in feedback. Questa mappa è una matrice dove al suo interno in ogni cella corrisponde un segnale di feedback associato (mostrato in figura 1).

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	3	60	38	47	84	21	50	36	24
3	32	96	74	100	98	90	25	87	82	95
4	40	68	78	58	80	97	52	71	26	39
5	22	16	72	76	14	37	37	37	37	13
6	34	69	62	81	46	37	37	37	37	9
7	92	11	70	33	37	2	2	2	37	66
8	91	54	51	63	37	2	2	2	37	20
9	35	30	28	33	37	37	37	37	37	57
10	45	6	7	17	8	53	61	73	85	10

Figura 1: Esempio casuale di mappa nella quale ad ogni cella corrisponde un segnale di feedback

Per ricostruire la mappa si è deciso di usare un algoritmo che si muove a spirale, andando a interrogare la matrice di adiacenza per poter collocare gli elementi nel giusto ordine. L'algoritmo ha l'obiettivo di individuare un modo per ricostruire una mappa che associa ai canali quantizzati la posizione del dispositivo, partendo da delle sequenze di canali quantizzati intercettati.
Per affrontare il problema, sono state assunte alcune semplificazioni: la prima è creare la matrice di adiacenza usando una mappa già creata, supponendo che le strisce fossero già state raccolte. La seconda invece riguarda l'assunzione che le sequenze di codici fossero esaurienti per la completezza della mappa. Nei prossimi paragrafi si spiega come questo problema sia stato affrontato partendo da una mappa casuale e tentando di ricostruirla tramite due algoritmi.

2.1 L'algoritmo 1

Nell'algoritmo 1 per ricostruire la mappa, si parte individuando i 4 potenziali angoli, e posizionandone uno nella prima cella in alto a sinistra (Figura 2). Da questo, si muove verso il basso cercando nella matrice di adiacenza gli elementi confinanti a quello esaminato precedentemente. Nel caso si trovasse una corrispondenza, l'elemento viene posto nella casella corretta. La procedura viene reiterata fino alla fine della colonna, ovvero quando non viene trovato un altro angolo (Figura 3). Una volta terminata la prima colonna, si ripete il processo per la riga inferiore, successivamente verso l'alto e poi verso sinistra fino al completamento della cornice esterna (Figura 4). Una volta arrivati al termine della prima procedura si cerca un altro angolo (ora più interno) e si ripete la procedura. Questo procedimento viene iterato fino a che la mappa ricostruita non è completamente riempita (Figura 5). L'algoritmo ora viene descritto nel dettaglio. Sono state realizzate due versioni, una semplificata e un'altra in cui la mappa può avere un cluster.

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Figura 2: Inserimento del primo angolo nella mappa ricostruita

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	77	0	0	0	0	0	0	0	0	0
3	32	0	0	0	0	0	0	0	0	0
4	40	0	0	0	0	0	0	0	0	0
5	22	0	0	0	0	0	0	0	0	0
6	34	0	0	0	0	0	0	0	0	0
7	92	0	0	0	0	0	0	0	0	0
8	91	0	0	0	0	0	0	0	0	0
9	35	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Figura 3: Completamento del primo lato

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	3	0	0	0	0	0	0	0	24
3	32	0	0	0	0	0	0	0	0	95
4	40	0	0	0	0	0	0	0	0	39
5	22	0	0	0	0	0	0	0	0	13
6	34	0	0	0	0	0	0	0	0	9
7	92	0	0	0	0	0	0	0	0	66
8	91	0	0	0	0	0	0	0	0	20
9	35	0	0	0	0	0	0	0	0	57
10	45	6	7	17	8	53	61	73	85	10

Figura 4: Completamento del bordo della mappa

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	96	60	38	47	84	21	50	36	24
3	32	96	96	100	98	90	25	87	82	95
4	40	68	96	96	96	96	96	71	37	39
5	22	16	72	76	96	37	37	37	37	13
6	34	69	62	81	96	96	64	94	65	9
7	92	11	70	33	56	79	31	19	2	66
8	91	54	33	33	37	5	49	37	4	20
9	35	30	33	33	93	48	37	15	18	57
10	45	6	7	17	8	53	61	73	85	10

Figura 5: Mappa completamente ricostruita

2.1.1 Versione semplificata

Si è preferito implementare prima una versione che prevede una semplificazione, ovvero che i codici siano univoci all'interno della mappa, avendo un rapporto uno ad uno tra il numero di elementi e il numero di celle.

Funzioni ausiliarie utilizzate

Nella scrittura dell'algoritmo sono state usate delle funzioni per gestire la ricostruzione della mappa, delle quali di seguito ne viene data una breve spiegazione:

1. **adjacency(R, C, mappa):** i parametri che vengono passati alla funzione sono una matrice, detta mappa, e le dimensioni della stessa, così da costruire una matrice di adiacenza. La mappa è riempita con dei numeri che vanno da uno al prodotto delle dimensioni: questi rappresentano i codici associati al precoder. La funzione restituisce una matrice che ha per dimensioni il prodotto delle dimensioni della mappa, dove ogni volta che un numero nella mappa risulta vicino ad un altro, viene incrementato il valore contenuto nella casella della matrice di adiacenza di uno.

2. **azzera(next):** la funzione prende come parametro un vettore di interi, e lo restituisce dopo aver sostituito ogni elemento con il numero zero.

3. **following(next, i, k):** la funzione prende come parametri il vettore nel quale vanno inseriti gli elementi vicini ad un codice già inserito nella mappa ricostruita, l'elemento analizzato e la variabile k, che è un contatore per posizionare l'elemento nella giusta cella del vettore. La funzione inserisce l'elemento i alla k - esima posizione.

4. **is present(e, rebuild):** la funzione prende come parametri un elemento e e la mappa ricostruita: ispezionando quest'ultima, restituisce il valore uno se l'elemento è presente, zero altrimenti.

Variabili ed oggetti presenti nell'algoritmo

Di seguito, vi sarano una breve introduzione di ogni matrice, vettore e variabile necessari alla scrittura dell'algoritmo e la descrizione dell'algoritmo stesso.

- **R e C:** queste due variabili, inizializzate all'inizio del codice, rappresentano il numero di righe e colonne della mappa
- **cont:** questa viene inizializzata a zero, e incrementata al completamento di ogni "cornice"; serve per sapere se il ciclo è al primo giro o meno.
- **conta elementi:** viene incrementata ad ogni inserimento di un elemento nella mappa ricostruita, così da tenere il conto di elementi inseriti
- **r e c:** queste due variabili vengono utilizzate per identificare la cella da riempire nella mappa ricostruita.
- **mr e mc:** vengono decrementate, permettendo all'algoritmo di arrestarsi al momento opportuno
- **prev:** questa varibile viene utilizzata per salvare l'ultimo elemento inserito nella mappa ricostruita
- **k:** è usata nella funzione following per tenere traccia della posizione in cui inserire gli elementi. Viene resettata ad uno ad ogni ciclo while.

- **S:** che contiene la somma di ogni colonna della matrice di adiacenza. Ciò comporta che gli indici del vettore corrispondono agli elementi della mappa

- **next:** contiene i plausibili elementi da inserire nella mappa

- **mappa:** la matrice mappa ha un numero di elementi pari ad R*C e viene inizialmente riempita con la funzione randperm(R*C) per poi essere riarrangiata in forma di mappa

- **adj:** la matrice viene dichiarata ed inizializzata grazie alla funzione adjacency, già spiegata in precedenza

- **adj copy:** questa è una copia della matrice adj, che verrà modificata nel corso dell'algoritmo mantenendo intatta la versione di adj originaria.

- **adj2:** Matrice di controllo per verificare se l'algoritmo ha ricostruito la mappa correttamente;

- **rebuild:** Questa è una matrice di zeri di dimensione R*C nella quale attraverso delle manipolazioni verrà ricreata la mappa.

2.1.2 Versione con un cluster

Il passaggio successivo prevede il caso in cui i codici nella mappa possono ripetersi in celle adiacenti dando luogo a dei "cluster" con lo stesso codice, in rappresentanza di celle che coprono un'area più vasta. Questo nuovo approccio ha portato ad una nuova versione dell'algoritmo, in grado di funzionare sia in delle mappe che contengono delle macchie di codici, sia nella versione ristretta affrontarta nei paragrafi precedenti. Nell'implementare questo algoritmo, è stato supposto che i codici ripetuti non potessero essere nella cornice;

questo ha dato la possibilità di riutilizzare una parte del codice precedente, assieme alle varie funzioni ausiliari.
L'algoritmo funziona tramite l'ancoraggio agli elementi circostanti. Una volta ricostruita la cornice, l'algoritmo interroga per ogni cella le quattro celle adiacenti che sono già state riempite per poi posizionare l'elemento che risulta adiacente a tutte e quattro.

0	0	0	0	0
0	0	0	0	0
33	37	0	0	0
33	93	48	37	15
17	8	53	61	73

Figura 6: Vengono interrogati i 4 elementi gia posizionati (37, 48, 93) per poter posizionare l'elemento corretto nella cella evidenziata

Questo codice può presentare degli errori nella ricostruzione della mappa. Questa discrepanza è dovuta al fatto che il codice analizza le sequenze in ordine crescente: ciò comporta che nel ricostruire un cluster che si sviluppa su più righe, può incorrere in delle incongruenze, nel caso in cui un codice della seconda riga della macchia fosse maggiore del codice nella cella adiacente: quest'ultimo codice sarebbe posizionato per primo. Quest'errore comporta un arresto della ricostruzione e l'ingresso dell'algoritmo in un ciclo infinito.

Variabili aggiuntive

L'algoritmo per risolvere il problema spiegato nel paragrafo precedente, sfrutta molte funzioni, vettori, matrici e variabili della versione che non prevedeva delle ripetizioni della mappa. In aggiunta, utilizza:

- **noa:** un vettore chiamato noa, Number Of Appearances, di lunghezza R*C, che per ogni elemento della mappa, rappresentato dall'indice delle

sue celle, ha salvato il numero di volte che questo deve comparire nella mappa. Questo è ottenuto attraverso l'uso del vettore S;

- **diag sx, left, above, diag dx:** queste quattro variabili memorizzano ad ogni ciclo gli elementi attorno alla cella che sta per essere riempita, con gli elementi già posizionati. Il nome è esplicativo, ma prendono gli elementi delle celle nelle due diagonali in alto, nella cella a sinistra e in quella sopra. Nel caso i 4 valori corrispondessero, si andrà a prendere i valori salvati nelle celle distanti di una posizione nella stessa direzione

2.2 Algoritmo 2

L'algoritmo 2 che viene proposto ora è una rivisitazione dell'algoritmo 1 dove vengono sostituite alcune variabili e ne vengono aggiunte di nuove, assieme a delle nuove funzioni. Il funzionamento generale è il medesimo, ma sono state effettuate alcune modifiche per adattarlo a risolvere un range molto più ampio di casi possibili. Nei prossimi paragrafi si andranno a vedere nel dettaglio tutte le modifiche e le aggiunte fatte all'algoritmo 1 per ottenere l'algoritmo 2.

2.2.1 Risoluzione delle precedenti problematiche

Il problema maggiore nell'algoritmo 1 è cadere in un loop infinito nel caso non fosse stato possibile posizionare correttamente uno o più valori all'interno della mappa. Per impedire il blocco di tutto l'algoritmo e di riuscire ad ottenere una mappa il più completa possibile si è deciso di cambiare approccio al posizionamento errato dei valori all'interno della mappa.
In questa versione l'algoritmo si divide in due fasi principali, il riempimento

della mappa e la risoluzione delle incertezze. Di seguito si andrà a trattare ogni fase nel dettaglio.

Riempimento della mappa

Il meccanismo di riempimento della mappa è il medesimo dell'algoritmo 1. Viene inserita una cella a partire da un angolo e si percorre a spirale tutto il bordo della matrice fino ad arrivare al centro. Nell'algoritmo 2 vengono resi più efficenti i cambiamenti delle variabili e delle matrici di controllo.
Il maggiore cambiamento è il controllo del valore da posizionare nella cella esaminata. Viene gestito tutto dalla funzione MainControl, spiegata in seguito, dove i risultati possono essere solamente due. Nel caso di certezza del valore da inserire, il controllo viene superato e viene memorizzato il valore corrispondente. Nel caso in cui non si dispone di abbastanza informazioni per verificare la certezza del valore da inserire, il controllo non viene superato e si inserisce, temporaneamente, un valore incerto denominato -1 per comodità. È possibile un terzo caso, più raro, in cui più di un valore supera tutto il controllo. Per evitare di inserire valori scorretti, anche questa situazione viene gestita inserendo un valore incerto e si lascia la gestione del caso alla seconda parte dell'algoritmo.
La prima fase termina quando tutte le celle hanno un valore assegnato, o il loro valore corretto o un valore incerto. Questa è la prima versione ricostruita grezza della mappa.

Risoluzione delle incertezze

Dopo aver assegnato un valore ad ogni cella, l'algoritmo verifica, con un controllo più accurato, quale valore inserire al posto del valore incerto situato nella cella esaminata. Si segue un procedimento simile alla fase preceden-

te. Si parte dall'angolo in alto a sinistra e con un movimento a spirale si vanno a selezionare le celle con un valore incerto. In questo modo si vanno a selezionare prima le celle situate nel bordo del cluster di valori incerti e successivamente il suo interno. Poiché una cella con valore -1 circondata da celle con valori uguali è irrisolvibile, si procede ripetendo tutto il percorso a spirale anche più di una volta in modo tale da analizzare sempre le celle del bordo fino a quando è possibile risolverne l'incertezza. Nei casi in cui o non si registrino più cambiamenti o i valori incerti siano terminati, si procede al controllo finale della matrice ricostruita con la matrice iniziale.
Il controllo che viene effettuato nella cella incerta consiste nell'osservare tutte e 8 le celle adiacenti ad essa, ricavare tutti i valori adiacenti ed effetuare un'intersezione tra tutti quei valori, in modo tale da trovare il valore comune a tutti.
Gli esiti anche in questa situazione sono 3:

1. Viene trovato il valore corretto e viene restituito in output dal controllo, così da poterlo inserire al posto del valore incerto;

2. L'intersezione produce un insieme vuoto. Il valore mantenuto nella cella rimane -1 fino ad ulteriori controlli;

3. L'intersezione produce più di un valore corretto. Poiché non si possono avere informazioni più accurate nella scelta dei valori, si è deciso di lasciare il valore incerto -1 e di proseguire con la risoluzione del resto della mappa.

Miglioramenti complessivi e adattabilità a più cluster

L'algoritmo è stato sviluppato in modo tale da non rimanere incompleto durante il suo svolgimento ed anzi è in grado di ricostruire la mappa nella

maniera più precisa possibile nonostante ci siano delle celle incognite. Viene inoltre risolto il problema della mappa di valori in cui sia presente un singolo cluster di valori poiché per conoscere il valore corretto da inserire è necessario conoscere anche solo una delle 8 celle adiacenti che abbia un valore diverso dal cluster. Tuttavia, l'unico caso, più raro, che non viene risolto correttamente è quello in cui una singola cella con un valore unico sia circondata da valori del cluster. In questo caso dato che la singola cella non risulta così grande nel quadro complessivo della mappa, il valore contenuto in essa verrà approssimato al valore delle celle del cluster.
Un ulteriore miglioramento è quello di poter avere molteplici cluster con valori differenti nella mappa e riuscire a risolverli in alcune determinate condizioni. Bisogna fare una distinzione nel posizionamento dei cluster poiché alcuni sono risolvibili mentre altri non permettono di avere abbastanza informazioni durante la ricostruzione da poter posizionare correttamente i valori di ogni cluster nelle celle corrispondenti. Sono stati individuati i seguenti casi:

- **I cluster sono tutti separati tra loro:** Nella situazione più favorevole, i cluster sono tutti separati tra loro. Si può trattare il caso come un analogo della mappa con all'interno un singolo cluster;

- **I cluster hanno una o due celle vicine adiacenti tra di loro:** Anche questo è un caso analogo a quello sopra riportato poiché sono presenti ancora abbastanza informazioni da poter ricostruire per intero la mappa;

- **3 o più celle dei cluster sono adiacenti tra di loro:** In questo caso l'algoritmo è in grado di conoscere il bordo complessivo dell'unione dei 2 o più cluster adiacenti. Tuttavia non è possibile conoscere il

posizionamento di ogni singola cella dei cluster all'interno del bordo calcolato dato che non ci sono abbastanza informazioni disponibili;

- **Un cluster si trova inglobato, in parte o per intero, in un altro cluster:** Per le motivazioni analoghe riportate nel caso precedente non è possibile dedurre la posizione di tutte le celle dei cluster. È comunque possibile delimitare il perimetro complessivo dei cluster in modo tale da riconoscere la zona incerta.

Di seguito verranno riportate tutte le variabili e le funzioni che sono state aggiunte, eliminando alcune delle variabili precedenti o migliorandole, fornendo maggiore efficacia nella risoluzione dell'algoritmo.

Le variabili aggiunte sono le seguenti:

- **adj lati:** Matrice copia di adj. Viene modificata durante l'esecuzione dell'algoritmo in maniera differente dalle modifiche applicate alla matrice adj copy per tenere traccia solo di alcuni cambiamenti specifici;

- **adjdiag:** Matrice che tiene conto solo degli elementi adiacenti posti nelle diagonali di una particolare cella;

- **control:** Variabile booleana utilizzata nel controllo principale durante il posizionamento di un valore in una cella;

- **cVector:** Vettore a dimensione variabile contenente i valori possibili da inserire in una cella.

- **fase:** Variabile intera che indica in quale parte dell'algoritmo ci si trova in modo tale da effettuare il controllo corretto delle celle adiacenti durante il posizionamento di un valore;

- **changes:** Variabile booleana che registra se ci sono stati cambiamenti nella fase finale della ricostruzione della mappa;
- **negCounter:** Variabile che conta il numero di celle in cui il valore non è ancora definito perché non sono presenti al momento sufficenti informazioni;
- **prev2:** Variabile che registra il penultimo valore inserito nella mappa. Utilizzata per mantenere la coerenza nella quarta sezione dell'algoritmo, prima dell'inizio di un nuovo ciclo.

Descrizone delle funzioni ausiliarie aggiuntive:

- **adjdecrease(mappa,SideMatrix,index, r,c):** La funzione viene chiamata durante la ricostruzione della parte interna della mappa all'interno di un ciclo for. Questa prende come parametri la matrice principale rebuild, la matrice adj_copy, l'indice corrente del ciclo for e la posizione corrente nella mappa tramite le variabili r e c. La funzione verifica una corrispondenza tra la SideMatrix e la MainMatrix tramite un ciclo for ed una condizione if. Quando la condizione diventa vera, viene decrementato di 1 il valore nella cella corrispondente nella SideMatrix;
- **adjacencydiag(R,C,mappa):** Questa funzione ha lo scopo di creare una matrice simile ad adj, ma i valori settati ad 1 sono solo le adiacenze poste sulle diagonali del valore in questione. La matrice non viene mai modificata durante tutto il programma e viene utilizzata per effettuare alcuni controlli sul posizionamento corretto deli valori nelle celle;
- **MainControl(mappa,adjMatrix,adjCopyMatrix,MatrixDiag,fase,iprev,...):** Funzione principale di tutto l'algoritmo. Vengono effettuati dei control-

li nelle celle adiacenti, già posizionate, alla cella analizzata verificando se queste sono adiacenti al valore i. Se tutte le celle sono adiacenti ad un dato valore i e se questo valore deve essere ancora posizionato, allora viene restituito in output il valore true. Nel caso una delle condizioni precedenti risulti falsa o uno o più valori adiacenti risulta -1 (quindi incerto), automaticamente il valore restituito è false;

- **negBorder(mappa,r,c):** Funzione di controllo che verifica se il valore incerto preso in considerazione sitrova nel bordo del cluster di valori incerti oppure al suo interno. Viene utilizzata nella fase finale dell'algoritmo per saltare automaticamente le celle in cui momentaneamente non è possibile calcolare il valore corretto per mancanza di informazioni;

- **selection(mappa,matrixLati,matrixDiag,r,c):** Funzione principale nella seconda parte dell'algoritmo. Viene fatto un controllo specifico su tutte e 8 le celle adiacenti alla cella presa in considerazione. Vengono esclusi automaticamente i valori incerti, mentre per i valori già posizionati viene effettuato un controllo per verificare quali valori risultano adiacenti ad esso. Viene poi fatta un'intersezione con tutti i valori comuni di tutte le celle adiacenti. Gli output possibili sono due. Nel caso in cui l'intersezione porti ad avere un solo valore, questo è anche il valore in output che verrà inserito nella cella. Nel caso in cui non ci siano valori o ce ne sia più di uno allora il valore in output è -1 e non si è riusciti a risolvere l'incertezza per quella determinata cella;

- **angleSelection(mappa,MatrixDiag,MatrixLati,r,c):** Funzione che applica gli stessi controlli utilizzati in selection ma vengono applicati alle 5 celle già posizionate. Il funzionamento è il medesimo.

Capitolo 3

Prestazioni degli algoritmi

Di seguito verranno descritte le prestazioni degli algoritmi e verranno mostrati alcuni casi particolari di mappe che possono essere risolti solo da alcuni algoritmi o da nessuno.

3.1 Casi particolari

Poiché le due versioni dell'algoritmo 1, per come sono state scritte, si interrompono durante l'esecuzione, verranno mostrati esempi di mappe prodotte in output dall'algoritmo 2 e commentate dalla prospettiva di tutti gli algoritmi.

3.1.1 Un cluster sviluppato su più righe e colonne

Partendo dal caso più semplice, nella figura 1 è presente una mappa contenente un cluster di valori sviluppato su più righe e colonne. La versione semplificata dell'algoritmo 1 non può risolvere alcun algoritmo con cluster perché per funzionare sono necessari solamente valori univoci. Questo sarà

valido anche per tutti i futuri casi particolari. La versione completa dell'algoritmo 1 è in grado di risolvere mappe contenenti cluster, ma presenta una limitazione. Se il cluster si sviluppa in più righe e/o colonne, i controlli che decidono il valore corretto da inserire cadono in un loop infinito, dovendo forzare manualmente l'interruzione dell'algoritmo. L'algoritmo 2 in questo caso non presenta problemi nella ricostruzione della mappa produce l'output corretto.

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	3	60	38	47	84	21	50	36	24
3	32	96	74	100	98	90	25	87	82	95
4	40	68	78	58	80	97	37	71	26	39
5	22	16	72	76	14	37	37	37	37	13
6	34	69	62	81	46	59	64	37	65	9
7	92	11	70	89	56	79	31	19	2	66
8	91	54	51	63	42	5	49	44	4	20
9	35	30	33	28	93	48	27	15	18	57
10	45	6	7	17	8	53	61	73	85	10

Figura 1: Mappa con presente un singolo cluster di valori quantizzati di feedback (37)

L'algoritmo 2 tuttavia non è infallibile nella ricostruzione di mappe con un singolo cluster poiché se il cluster è abbastanza articolato, alcuni controlli non possono essere certi al 100%. È il caso nella figura 2 in cui i controlli effettuati sulle celle incerte (-1), non assicurano al 100% di inserire l'elemento corretto e si è preferito lasciare incerta la cella stessa.

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	3	60	38	47	84	21	50	36	24
3	32	96	74	100	98	90	25	87	82	95
4	40	68	78	58	80	97	52	71	37	39
5	22	16	72	76	14	37	37	37	37	13
6	34	69	62	81	46	59	-1	37	65	9
7	92	11	70	33	56	-1	-1	37	2	66
8	91	54	51	63	37	5	49	37	4	20
9	35	30	28	33	93	48	37	15	18	57
10	45	6	7	17	8	53	61	73	85	10

Figura 2: Mappa con presente un singolo cluster articolato

3.1.2 Due o più cluster

Aumentando il numero di cluster entrambe le versioni dell'algoritmo 1 non funzionano, bloccandosi o entrando in loop. C'è qualche caso in cui, se i cluster si sviluppano in una riga e sono tutti separati tra loro, la versione completa potrebbe funzionare, ma per tutti gli altri casi l'algoritmo 1 non funziona. Nella figura 3 è presente una mappa irrisolvibile per le versioni dell'algoritmo 1 poiché i controlli da effettuare non sono sufficienti ad inserire il valore corretto. Questo caso rientra invece nei casi risolvibili dell'algoritmo 2, che riesce a produrre correttamente in output la mappa.

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	96	60	38	47	84	21	50	36	24
3	32	96	96	100	98	90	25	87	82	95
4	40	68	96	96	96	96	96	71	37	39
5	22	16	72	76	96	37	37	37	37	13
6	34	69	62	81	96	96	64	94	65	9
7	92	11	70	33	56	79	31	19	2	66
8	91	54	33	33	37	5	49	37	4	20
9	35	30	33	33	93	48	37	15	18	57
10	45	6	7	17	8	53	61	73	85	10

Figura 3: Mappa in cui sono presenti molteplici cluster sviluppati

Di seguito nelle figure 4 e 5 sono riportati i casi in cui nessuno degli al-

goritmi è in grado di risolvere.

Nella prima mappa sono presenti due cluster adiacenti tra loro. L'algoritmo 2 è in grado di definire il bordo complessivo dei due cluster perché possiede ancora le informazioni necessarie per effettuare correttamente i controlli. Tuttavia superato il bordo dei cluster non è possibile conoscere più nulla perché non ci sono informazioni certe per poter collocare correttamente una cella di un cluster o dell'altro.

La seconda mappa è il caso con maggiore perdita di informazione possibile. Sono presenti due cluster in cui uno viene circondato dall'altro, ma si può notare che il cluster più piccolo non può essere definito in alcun modo. L'algoritmo 2 è in grado di conoscere il valore di feedback quantizzato che si trova dentro il cluster più grande, ma non può sapere il suo posizionamento.

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	3	60	38	47	84	21	50	36	24
3	32	96	74	100	98	90	25	87	82	95
4	40	68	78	58	80	97	52	71	26	39
5	22	16	72	76	14	37	37	37	37	13
6	34	69	62	81	46	37	-1	-1	37	9
7	92	11	70	33	56	2	-1	-1	2	66
8	91	54	51	63	42	2	2	2	2	20
9	35	30	28	33	93	48	27	15	18	57
10	45	6	7	17	8	53	61	73	85	10

Figura 4: Mappa in cui sono presenti due cluster adiacenti tra di loro

	1	2	3	4	5	6	7	8	9	10
1	1	99	86	41	67	29	88	12	75	55
2	77	3	60	38	47	84	21	50	36	24
3	32	96	74	100	98	90	25	87	82	95
4	40	68	78	58	80	97	52	71	26	39
5	22	16	72	76	14	37	37	37	37	13
6	34	69	62	81	46	37	-1	-1	37	9
7	92	11	70	33	37	-1	-1	-1	37	66
8	91	54	51	63	37	-1	-1	-1	37	20
9	35	30	28	33	37	37	37	37	37	57
10	45	6	7	17	8	53	61	73	85	10

Figura 5: Mappa in cui un cluster si trova all'interno dell'altro

3.2 Tabella delle prestazioni

Mappa	Algoritmo1s	Algoritmo1c	Algoritmo2
A valori univoci	X	X	X
Con un cluster	O	/	X
Con un cluster complesso	O	O	/
Con più cluster separati	O	O	/
Con più cluster adiacenti/inglobati	O	O	/

Tabella 3.1: X = Ricostruisce completamente la mappa; / = Ricostruisce parzialmente la mappa; O = Non ricostruisce mai la mappa; 1s = 1semplificato; 1c = 1completo.

Capitolo 4

Conclusioni

Si è cercato con questa tesi di trovare un metodo di ricostruzione efficace di una mappa che rappresenta la copertura di una stazione base. Sono stati analizzati molteplici scenari possibili di mappa e sono stati testati due algoritmi risolutivi con prestazioni differenti per trovare quello che riesca a ricostruire la mappa con la maggiore affidabilità possibile.
Tutti gli algoritmi hanno un funzionamento generale simile, ma differiscono per la quantità di controlli che effettuano e per la gestione degli errori. Dopo aver testato le prestazioni degli algoritmi si è notato che l'algoritmo 2 riesce a performare meglio di tutti, riuscendo a ricostruire quasi sempre buona parte o tutta la mappa. Tuttavia non c'è nessun algoritmo, con le informazioni a disposizione, che riesca a ricostruire sempre al 100% dei casi una mappa partendo solo da sequenze di segnali di feedback.

Bibliografia

[1] Roth, S., Tomasin, S., Maso, M., Sezgin, A. (2021). Localization attack by precoder feedback overhearing in 5G networks and countermeasures. IEEE Transactions on Wireless Communications, 20(7), 4100-4112.