Controllo Multi-Agente per Stima Distribuita

Adriano Salata

Abstract

Questa relazione presenta un algoritmo distruibito per il controllo di un sistema multiagente, al fine di stimare un campo di interesse all'interno di un ambiente sconosciuto. Una logica conseguenza del fatto che si hanno a disposizione piu dispositivi per l'esplorazione, è quella di diminuire il carico di lavoro compiendo un partizionamento e assegnando ad ogni robot una specifica regione di competenza in cui effettuare le misurazioni. Si presenta, quindi, un algoritmo di partitioning, grazie al quale si minimizzano le distanze da percorrere per raggiungere ogni punto della mappa. La seconda parte fondamentale del progetto consiste nello stimare il campo di interesse, si definisce un algoritmo di stima distribuita, grazie al quale ogni robot condivide con gli altri le informazioni apprese dall'area assegnatagli, ottenendo, in un numero finito di iterazioni, la convergenza ad una rappresentazione globale del campo.

Introduzione

Recentemente, si è visto un interesse crescente per le reti wireless di sensori [1], dove un insieme di nodi in grado di effettuare misurazioni e comunicare possono monitorare ambienti, il controllo del traffico, sistemi di sorveglianza e l'automazione industriale [2]. Intorducendo in questi sistemi la presenza di sensori mobili, inoltre, si ottengono prestazioni decisamente superiori, soprattutto per quanto riguarda l'adattabilità ad ambienti dinamici [3]. Ogni agente ha delle capacità limitate, sia per quanto riguarda il raggio di comunicazione, sia per quanto riguarda la potenza di calcolo, ma in gruppo, sono in grado di compiere varie operazioni, come l'esplorazione, la sorveglianza e il monitoraggio di ambienti ([4], [5], [6], [7] e [8]). Una applicazione molto interessante di questo tipo di sistemi è quello dell'inseguimento dei picchi di salita del gradiente del campo che si sta misurando. Un esempio di questo tipo di approccio è il monitoraggio di tossine all'interno un ambiente acquatico come il mare o un lago, in cui si rileva la presenza di particolari tipi di alghe che rilasciano una sostanza detta cianotossina, in grado di provocare grossi danni all'uomo,alla fauna e alla flora. Misurando il livello di clorofilla nell'acqua, e spostando i sensori nei punti in cui diviene più densa, è possibile stimare come queste alghe si distribuiscano nell'ambiente. L'approccio più comune a questo tipo di problemi è quello biologico della ([9], [10]), in cui i corpi cellulari (batteri o altri organismi) si spostano verso le concentrazioni massime di molecole nutrienti. In questo modo, però, il rate di convergenza è piuttosto basso e vi è il pericolo che gli agenti si blocchino in un massimo locale del campo. Le rete cooperative di agenti mobili sono presentate in [11] e [12] in cui si mostra che queste reti sono in grado di adattare la propria configurazione all'ambiente in questione, in modo da ottimizzare la stima e l'inseguimento del gradiente. Il coordinamento tra più individui si presenta spesso anche nel mondo animale, ad esempio i banchi di pesce si muovono in maniera coordinata nelle zone in cui il gradiente della densità di cibo aumenta ([13],[14]), mentre, negli stormi di uccelli, i singoli individui comunicano con quelli vicini in modo da ottenere un movimento collettivo senza scontrarsi tra loro. In questo progetto, si applica un algoritmo di stima distribuita e controllo cooperativo che mima il comportamento individuale e globale di un gruppo di animali che comunicano le informazioni locali apprese, in modo da raggiungere un obbiettivo comune. Si descrive inizialmente un controllo in cui ogni agente utilizza esclusivamente le proprie misure per ottenere una rappresentazione del campo di interesse, successivamente si introduce un termine di consenso in modo che ogni robot propaghi nella rete le proprie misure permettendo anche agli altri agenti di utilizzarle per ottenere una stima più accurata.

Un importante aspetto di cui tenere conto è come conviene che i vari agenti si distribuiscano nell'ambiente, in modo tale da ottenere una convergenza al risultato finale più velocemente possibile. Problematiche di questo tipo si trovano in svariati contesti, ad esempio, in applicazioni di sorveglianza, ad ogni sensore viene assegnata una regione di competenza, in modo da assicurare che tutta l'area sia monitorata correttamente. Altro esempio sono i servizi di trasporto o consegna, in cui si cerca di minimizzare il tempo di attesa dei clienti suddividendo l'area da coprire in più zone, ad ognuna delle quali viene assegnato un particolare veicolo. In letteratura esistono vari algoritmi atti a risolvere questo tipo di problematiche, in generale ognuno di essi si basa sul cercare di minimizzare la spazio che ogni agente deve percorrere per raggiungere i punti dell'ambiente in questione, eventualmente pesando alcune zone con un qualche criterio di priorità ([15], [16]). Per avere una certa scalabilità del problema, cioè per poter applicare gli algoritmi indipendentemente dalle dimensioni dell'ambiente da esplorare e dal numero di agenti utilizzati, viene fatta un'importante ipotesi. Si suppone che gli agenti utilizzati siano omogenei, cioè dotati degli stessi sensori, che gli conferiscono la capacità di effettuare delle misure sul segnale e di rilevare eventuali ostacoli. In questo modo e sufficiente aumentare il numero di robot impiegati per poter ampliare il raggio d'azione complessivo. Per ottenere un partizionamento ottimo dell'ambiente, si deve introdurre ancora una volta un termine di consenso, cioè ogni agente calcola la propria area di competenza sulla base delle infor-

mazioni locali e condivide il risultato ottenuto con i robot vicini. In questo modo, si ha la convergenza ad una configurazione globale che minimizza i tempi di spostamento all'interno di tutto l'ambiente.

L'ultimo aspetto di cui tenere conto è l'errore che si commette nella stima del campo di interesse. Utilizzando un approccio di tipo distribuito, infatti, si ha un degradamento della stima, ed è necessario verificare se l'errore commesso rispetto alla stima ottenuta in maniera centralizzata non superi dei limiti prefissati. Un importante strumento utilizzato per verificare la bontà del modello creato è il metodo della Cross Validation (CV), che, in base alle misure a disposizione, permette di utilizzare parte di esse per testare la capacità di predizione del modello. In letteratura esistono numerose implementazioni di questo particolare tipo di metodo ([17], [18] e [19]), inoltre è possibile trovare alcune applicazioni anche in ambito distribuito [20]. In questo progetto, però, si utilizza la CV in maniera distribuita per regolarizzare il problema di stima nelle situazioni in cui è mal condizionato (ill-posed problem) e per verificare in quali condizioni si ottiene un errore minimo.

Contributo

In letteratura sono già presenti alcune soluzioni per risolvere separatamente i problemi di partitioning e di stima distribuita. In questo progetto, però, si presenta una soluzione alla stima distribuita con un sistema multiagente, in modo tale che queste due operazioni vengano svolte in maniera simultanea e, soprattutto, in modo tale da avere una collaborazione tra i due algoritmi, per avere una stima del campo di interesse che venga migliorata ogni volta che i vari agenti acquisiscono una misura. Sfruttando il valore stimato, è poi possibile modificare continuamente il partizionamento e concentrare le osservazioni in quei punti che si ritengono maggiormente significativi, come le aree in cui il campo di interesse assume valori maggiori, oppure quelle in cui varia più velocemente ed è più difficoltoso ottenere una stima con errore di predizione basso.

Nella parte relativa al partitioning, ci si è basati sulle soluzioni presenti in letteratura [16], sfruttando alcuni dei teoremi e degli algoritmi descritti, ma è stato necessario andare a modificare alcune delle dimostrazioni e degli algoritmi presenti, in modo tale da poterli applicare nella soluzione implementata ed in modo tale da poterli integrare con la stima del campo di interesse.

Infine, un contributo dato da questo progetto che non è già presente in letteratura, è l'utilizzo di algoritmi distribuiti per regolarizzare il problema nel caso in cui risulti essere mal condizionato, sfruttando anche metodi di validazione per verificare la capacità di predizione del modello ottenuto tramite la stima distribuita.

Sommario

Questa relazione si divide in due parti fondamentali. La sezione 1 descrive partitioning della mappa, in cui vengono descritti vari approcci alla soluzione di questo tipo di problemi, ed, in particolare, si propone un algoritmo di consensus in grado di ottenere una suddivisione ottima dello spazio sfruttando le informazioni locali possedute da ogni agente. La sezione 2 è incentrata sulla stima distribuita del campo di interesse. nella sezione 2.1.1 viene brevemente descritto un algoritmo di stima ai minimi quadrati, prendendo in considerazione sia una implementazione centralizzata, utilizzando un solo agente, sia una implementazione distribuita che sfrutta le informazioni acquisite da più agenti. Nella sezione 2.2 è stata data particolare attenzione ai problemi di regolarizzazione in situazione di singolarità, effettuando una trattazione degli algoritmi applicabili sia in ambito centralizzato che in ambito distribuito, e confrontando i risultati ottenuti nei due casi. La sezione 3 descrive come gli algoritmi creati per la soluzione dei problemi di partitioning e di stima distribuita possano collaborare per ottenere un aggiornamento continuo sia della stima sia del partizionamento ogni volta che gli agenti acquisiscono una nuova misura.

1 Partitioning

1.1 Formulazione del problema

Consideriamo un gruppo di agenti robotici con limitate capacità di calcolo e di comunicazione, e un ambiente sconosciuto da esplorare. Vogliamo suddividere tale ambiente in regioni più piccole, ognuna delle quali verrà poi assegnata ad un singolo agente. Il nostro approccio sarà quello di aggiornare in maniera iterativa le partizioni, in modo da minimizzare una funzione costo che descriva lo spazio che ogni agente deve percorrere per eplorare i punti dell'area che gli compete.

Denotiamo con Q l'ambiente in questione, che assumiamo inizialmente come un sottoinsieme compatto e convesso di $\mathbb{R}^p$, le partizioni di Q sono definite in questo modo:

Definition 1 *(N-partizione)* Una N-partizione di Q, $V = (V_i)_{i=1}^N$è una successione ordinata di sottoinsiemi di Q con le seguenti proprietà:

1. $\bigcup_{i=1}^N V_i = Q$;
2. $int(V_i) \bigcap int(V_j) = \emptyset \ \forall i \neq j$;
3. ogni sottoinsieme v_i è chiuso ed ha interno non vuoto.

Sia $p := (p_1, ..., p_N) \in Q^N$ l'insieme contenente le posizioni degli N agenti all'interno dell'ambiente Q. Si crea una corrispondenza biunivoca tra tale insieme e la partizione

v, infatti, ad ogni agente viene assegnata una sola regione di competenza V_i e viceversa.

Definiamo ora una funzione crescente $f : \mathbb{R}_+ \to \mathbb{R}_+$, che dipende dalla distanza tra i punti di Q e una funzione peso $\phi : Q \to \mathbb{R}_+$, che permette di dare una priorità diversa ai vari punti da esplorare. Per ottenere un partizionamento che ottimizzi le distanze che devono percorrere i vari agenti, tenendo conto del peso dato ad ogni punto di Q, si considera la funzione

$$H_{multicenter} := \sum_{i=1}^{N} \int_{V_i} f(\|p_i - q\|)\phi(q)dq \qquad (1)$$

l'obbiettivo è minimizzare tale funzione costo, che assume significati diversi in base a come vengono scelte f e ϕ. Ad esempio, nel sistema di trasporto e consegna descritto in precedenza, la funzione $f(\parallel p_i - q \parallel)$ potrebbe descrivere le distanze che il veicolo i-esimo deve percorrere per raggiungere i punti di Q, $\phi(q)$ potrebbe descivere la probabilità con cui il punto q deve essere raggiunto e $H_{multicenter}$ quantificherebbe il tempo che i vari clienti dovrebbero attendere per essere raggiunti. Nel caso preso in esame, la funzione f rimane quella vista in precedenza, mentre la funzione ϕ può essere scelta come la funzione stimata stessa, in modo da avere un maggior numero di sensori nei punti in cui assume valori maggiori, oppure il suo gradiente, in modo da avere più misurazioni nei punti in cui si hanno variazioni maggiori, e quindi maggiori errori di stima. Ovviamente, è possibile assegnare una priorità di questo tipo solo dopo che si è stimato il campo almeno una volta, quindi, inizialmente si è scelto di non pesare in maniera differente i punti della mappa.

1.1.1 Partizioni di Voronoi

Le partizioni ottime che si ottengono minimizzando la funzione costo vengono dette *partizioni di Voronoi*, e sono definite da:

$$V_i := \{q \in Q | \parallel q - p_i \parallel \leq \parallel q - p_j \parallel, \forall j \neq i\} \qquad (2)$$

in particolare, la funzione costo assume il valore minimo

$$\begin{aligned} H_{Voronoi} &= \int_Q \min_{i\in\{1,..,N\}} f(\parallel p_i - q \parallel)\phi(q)dq \qquad (3) \\ &= E_{(Q,\phi)}\left[\min_{i\in\{1,..,N\}} f(\parallel p_i - q \parallel)\right] \end{aligned}$$

per semplicità consideriamo che sia $f(\parallel p_i - q \parallel) := \parallel p_i - q \parallel^2$, e calcoliamo i *centroidi* di ogni partizione, cioè i punti che si trovano alla distanza minima rispetto ad ogni altro punto della partizione:

$$C(V_i) = \left(\int_{V_i} \phi(q)dq\right)^{-1} \int_{V_i} q\phi(q)dq \qquad (4)$$

che risulta essere l'unico punto che minimizzi la funzione $\int_{V_i} \parallel p_i - q \parallel^2 \phi(q)dq$. Riprendendo la funzione di Voronoi, e derivandola rispetto ad una delle posizioni p_i, si ottiene [21]

$$\frac{\partial H_{Voronoi}}{\partial p_i}(p) = 2\int_{V_i} q\phi(q)dq(p_i - C(V_i)) \qquad (5)$$

quindi il minimo locale della funzione di Voronoi, all'interno di ogni partizione, è ottenuto in corrispondenza del centroide della partizione stessa. Il problema si riduce a determinare il partizionamento che minimizzi la funzione costo calcolata nei vari centroidi. Esistono degli algoritmi che permettono di determinare tale configurazione, ad esempio l'algoritmo di Lloyd permette di minimizzare la funzione $H_{Voronoi}$ sia per sistemi a tempo continui sia per sistemi a tempo discreto, ed è possibile trovare una dimostrazione della sua convergenza in [22]. Questo algoritmo, però, richiede che ogni agente possa comunicare con tutti gli altri e che le comunicazioni avvengano in maniera sincrona, di conseguenza non è possibile applicarlo al nostro caso pratico, in cui il raggio di comunicazione è limitato e vi possono essere dei ritardi nella trasmissione. Perciò, si è deciso di utilizzare un modello di comunicazione di tipo *gossip*, in cui la comunicazione può avvenire esclusivamente tra agenti adiacenti.

1.1.2 Partizioni di Voronoi in ambiente discretizzato

Fino ad ora si è sempre supposto di avere un insieme Q convesso, nel quale, cioè, per ogni coppia di punti, il segmento che li congiunge è interamente contenuto nell'insieme stesso. Questa è una grossa limitazione per quanto riguarda la possibilità di applicare il partizionamento visto ad ambienti reali, ad esempio, se consideriamo l'esplorazione delle stanze di un'appartamento, vi possono essere dei punti non collegabili direttamente da un segmento.

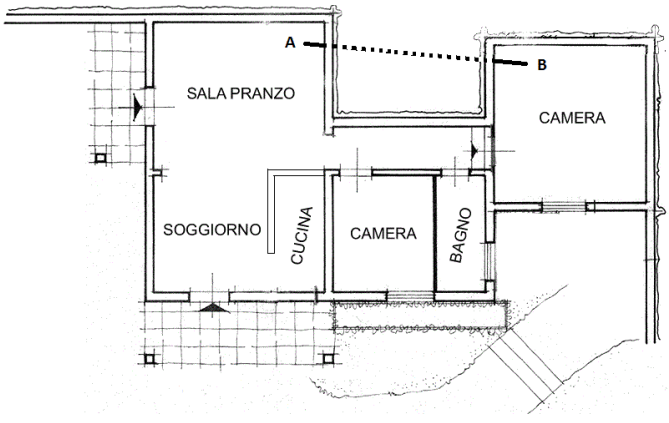

Figura 1: Mappa di un appartamento non assimilabile ad un insieme convesso.

Pertanto si è preferito effettuare una discretizzazione dell'ambiente, costruendo su di esso una griglia e assegnando ad ogni cella un valore che rappresenti la presenza o meno di un ostacolo. In [16] sono presenti parte delle definizioni e dei teoremi che seguono, ed è possibile trovare una discretizzazione simile, in cui si assegna ad ogni cella il vertice di un grafo e come distanza tra due vertici la

lunghezza del cammino minimo che li congiunge. Nel caso in esame, invece, si è preferito non introdurre un grafo e considerare come distanza tra due celle la distanza euclidea tra i loro punti centrali (in appendice è data una definizione di distanza tra due celle che tiene conto della presenza di ostacoli). É possibile estendere i risultati visti nella sezione precedente al caso discreto, si utilizza come funzione f la distanza tra due celle $d(h,k)$, o $d_{V_i}(h,k)$ se si vogliono considerare soltanto le celle di V_i. Il costo che un agente ha per esplorare l'area assegnatagli V_i a partire da una cella h

$$H_{one}(V_i,h) := \sum_{k\in V_i} d_{V_i}(h,k)\phi(k) \tag{6}$$

il centroide di tale area è la cella h che minimizza

$$C(V_i) = \arg\min_{h\in V_i} H_{one}(V_i,h) \tag{7}$$

se consideriamo un insieme di punti $c = \{c_1,...,c_n\} \in V$, con $c_i \in V_i$, otteniamo che la funzione $H_{multicenter}$ è data da

$$H_{multicenter}(V,c) = \sum_{i=1}^{N} H_{one}(V_i,c_i). \tag{8}$$

Anche in questo caso la disposizione ottima si ottiene in corrispondenza dei centroidi, si cercherà quindi di determinare la partizione che minimizza la funzione

$$H_{centroid}(V) = H_{multicenter}(V, C(V))$$

dove $C(V)$ rappresenta il vettore contenente i centroidi delle partizioni di V. Introduciamo ora alcuni concetti che permetteranno di ottenere un partizionamento ottimo per sistemi con comunicazione gossip [16].

Definition 2 *(Partizione di Voronoi generata da un punto)* Dato un vettore di punti distinti $c \in Q^N$, una partizione V di Q è definita *partizione di Voronoi di Q generata da c* se, per ogni V_i e per ogni $h \in V_i$ si verifica che $d(h,c_i) \leq d(h,c_j)$, $\forall\, j \neq i$.

Una importante proprietà della funzione costo, dimostrata in [16]:

Proposition 3 Data una partizione V di Q e $c \in Q^N$, se V' è una partizione di Voronoi generata da c e $c' \in Q^N$ è tale che $c_i \in C(V_i)\,\forall i$, allora

$$H_{multicenter}(V',c) \leq H_{multicenter}(V,c)$$

$$H_{multicenter}(V,c') \leq H_{multicenter}(V,c)$$

da ciò si deduce facilmente che (V,c) minimizza $H_{multicenter}(V,c)$ se e solo se $c_i \in C(V)$ e V è una partizione di Voronoi generata da c. A questo punto, fissato $c = C(V)$, dove V è una generica partizione di Q, e data una partizione di Voronoi V^*, vale la disuguaglianza:

$$H_{centroid}(V^*) \quad \leq \quad H_{centroid}(V). \tag{9}$$

Definition 4 *(Partizione di Voronoi Centroidale [16])* Una partizione V di Q è detta *partizione di Voronoi centroidale* di Q se esiste un $c \in Q^N$ tale che V sia una partizione di Voronoi generata da c e $c \in C(V)\,\forall i$.

Arriviamo a questo punto al partizionamento ottenuto tramite comunicazione gossip, introducendo il concetto di partizionamento ottimo a coppie, in cui, per ogni coppia di regioni adiacenti, non è possibile ottenere una 2-partizione migliore dell'unione delle aree stesse [16].

Definition 5 *(Partizione ottima a coppie)* Una partizione V di Q è una *partizione ottima a coppie (optimal pairwise partitioning)* di Q, se, per ogni coppia (i,j) di regioni adiacenti di Q si verifica che:

$$H_{one}(V_i,C(V_i)) + H_{one}(V_j,C(V_j)) = \min_{a,b\in V_i \bigcup V_j} \Gamma \tag{10}$$

$$\Gamma = \left\{ \sum_{h\in V_i \bigcup V_j} \left\{ min(d_{V_i \bigcup V_j}(a,h), d_{V_i \bigcup V_j}(b,h)) \right\} \phi(h) \right\} \tag{11}$$

a questo punto non rimane che da dimostrare che l'insieme delle partizioni ottime a coppie non è altro che un sottoinsieme di quello delle partizioni di Voronoi. La dimostrazione di questa proposizione è presente in [16], ma si suppone di aver associato alla rete un grafo connesso, ed è stato necessario apportare alcune modifiche per dimostrare che la proprietà descritta continua a valere anche nel contesto di questo progetto.

Proposition 6 Sia V una partizione ottima a coppie di Q, allora V è anche una partizione di Voronoi centroidale.

Proof Supponiamo per assurdo che V sia una partizione ottima a coppie di Q, ma che non sia una partizione di Voronoi centroidale. Allora esistono due componenti V_i e V_j di V e un elemento $x \in V_i$ tale che:

$$d(x,C(V_i)) > d(x,C(V_j)) \tag{12}$$

prendiamo V_j in modo tale che, per ogni $k \neq j$

$$d(x,C(V_k)) \geq d(x,C(V_j)) \tag{13}$$

consideriamo ora la più breve sequenza di celle che congiunge il punto x con il centroide di V_j, e sia m la prima cella di tale sequenza. Sia ora l tale che $m \in V_l$, allora vi sono due possibilità:

1) Se $m = x$, si ha che

$$d_{V_i}(x, C(V_i)) \geq d(x, C(V_i)) > d(x, C(V_j)) = d_{V_i \cup V_j}(x, C(V_j))$$

ciò è assurdo, poichè, siamo nelle ipotesi di partizione ottima a coppie.

2) Se $m \neq x$, dalla (13):

(a) $d(m, C(V_l)) > d(m, C(V_j))$

(b) $d(m, C(V_l)) = d(m, C(V_j))$

nel primo caso, tenendo conto del fatto che $m \in V_l$ e applicando lo stesso ragionamento del caso $x = m$, si trova l'assurdo. Nel secondo caso, è necessario verificare se esistono una successione di celle tra m e $C(V_l)$ tale che ognuna di esse appartenga a V_l. Se ciò non accade si ha che:

$$d_{V_l}(m, C(V_l) > d(m, C(V_l)) = d_{V_l \cup V_j}(m, C(V_j))$$

si presenta ancora una volta l'assurdo. Infine, se questa successione dovesse effettivamente esistere, si applica lo stesso ragionamento scegliendo V_l al posto di V_j. Dal momento che la distanza tra il centroide di V_l e m è uguale a quella tra il centroide di V_j e m, la disuguaglianza (12) è ancora verificata, ma si ha che la nuova cella m sarà più vicina ad x. Ripetendo un numero finito di volte il ragionamento, si troverà certamente un vertice che porti all'assurdo. ♢

come conseguenza di questo teorema, si ha che è possibile ottenere un partizionamento ottimo di Q effettuando esclusivamente confronti tra regioni adiacenti, ed è quindi attuabile nel contesto in cui ci siamo messi.

1.2 Rete di Agenti con Comunicazione Gossip

Come visto, si vuole ottenere una partizione ottima di Q attraverso degli agenti in grado di effettuare soltanto comunicazioni asincrone e a corto raggio. Una delle possibili soluzioni a questo tipo di problemi è utilizzare un algoritmo di tipo gossip, cioè un particolare tipo di algoritmo di consensus ([23],[24],[25]). L'idea dell'algoritmo del consensus è quello di trovare un punto d'accordo tra tutti gli agenti in gioco in un sistema dinamico rappresentabile dall'equazione di aggiornamento

$$x(t+1) = Px(t) \tag{14}$$

con P matrice stocastica che soddisfa le proprietà

$$P_{ij} \geq 0 \text{ e } \sum_j P_{ij} = 1$$

che equivale a

$$P\mathbf{1} = \mathbf{1}$$

si vuole far tendere x ad un vettore di componenti tutte uguali, cioè

$$x(t) \to \alpha\mathbf{1}$$

L'*algoritmo gossip simmetrico* è un algoritmo randomizzato in cui, ad ogni istante, due agenti adiacenti iniziano a comunicare e si scambiano informazioni riguardanti il proprio stato, calcolando il nuovo stato come una *combinazione convessa*

$$x_i(t+1) = (1-q)x_i(t) + qx_j(t) \tag{15}$$

$$x_j(t+1) = qx_i(t) + (1-q)x_j(t) \tag{16}$$

tutti gli altri non ricevono informazione

$$x_l(t+1) = x_l(t), \forall i \neq j \tag{17}$$

$$\tag{18}$$

P è casuale, dipende da quali agenti iniziano a comunicare, e le comunicazioni sono indipendenti, cioè ciò che accade all'istante t è indipendente da ciò che è accaduto nell'istante precedente. Non si è certi della convergenza dell'algoritmo, ma, in caso positivo, converge alla media delle condizioni iniziali (*average consensus*).

Sia in questa parte del progetto, si in quella relativa alla stima distribuita, si adotta proprio un algoritmo di questo tipo, in cui gli agenti, a due a due, aggiornano il proprio stato, compiendone la media aritmetica.

Una volta difinito il modo in cui gli agenti comunicano è necessario fare alcune ipotesi sulle loro proprietà:

1. l'agente *i-esimo* conosce esclusivamente il proprio identificatore, cioè i;
2. l'agente *i-esimo* è in grado di memorizzare le celle di Q e di compiere operazione su di esse;
3. l'agente *i-esimo* è in grado determinare la cella in cui si trova.

Remark 7 Per poter determinare la posizione dei vari agenti è necessario che ci sia un sistema GPS (Global Position System), in realtà questa proprietà può essere rilassata se si ha come unico obbiettivo quello di effettuare un partitioning dell'ambiente. In questo caso, infatti, è sufficiente che ogni agente sia in grado di calcolare il centroide della propria regione e di determinare la sua distanza dalle altre celle. Si rende però necessario, quando si desidera spostare l'agente nell'ambiente e effettuare le misurazioni, perchè, altrimenti, non sarebbe possibile associare la misura ad una coordinata nello spazio.

1.2.1 Algoritmo di Partitioning Gossip

L'obbiettivo è quello di implementare un algoritmo distribuito che aggiorni in maniera iterativa la partizione $V(t) = \{V_i(t)\}_{i=1}^N$, risolvendo il problema di ottimizzazione:

$$\arg\min_{V} H_{centroid}(V)$$

e permettendo la comunicazione di soli due agenti adiacenti per ogni iterazione. Vediamo ora un possibile algoritmo in pseudo codice, che verrà successivamente implementato. Ad ogni iterazione di questo algoritmo viene selezionata una coppia di agenti adiacenti in maniera casuale che esegue le seguenti operazioni per determinare la propria partizione:

---

function PAIRWISEPARTITIONING($V_i(t), V_j(t)$)
  $V_i^* \leftarrow V_i(t)$
  $V_j^* \leftarrow V_j(t)$
  $c_i \leftarrow C(V_i(t))$
  $c_j \leftarrow C(V_j(t))$
  $W \leftarrow V_i \cup V_j$
  S:= lista di tutte le coppie di celle di W
  for all (a, b) ∈ S **do**
    $V_a \leftarrow \{x \in W : d_W(x,a) \leq d_W(x,b)\}$
    $V_b \leftarrow \{x \in W : d_W(x,a) > d_W(x,b)\}$
    if $H_{one}(a, V_a) + H_{one}(b, V_b) <$
      $H_{one}(c_i, V_i^*) + H_{one}(c_j, V_j^*)$ **then**
        $V_i^* \leftarrow V_a$
        $V_j^* \leftarrow V_b$
        $c_i \leftarrow a$
        $c_j \leftarrow b$
    $V_i(t+1) \leftarrow V_i^*$
    $V_j(t+1) \leftarrow V_j^*$

Remark 8 Nell'algoritmo viene assegnato V_a all'agente i e V_b all'agente j, senza controllare se le nuove regioni corrispondano effettivamente a quelle precedenti. Questo può portare ad uno scambio tra le posizioni dei due agenti, nell'implementazione si è preferito gestire in maniera leggermente diversa la determinazione delle nuove aree, in modo da evitare tali scambi.

1.3 Analisi della Convergenza

Vediamo ora alcune proposizioni necessarie a dimostrare la convergenza dell'algoritmo appena descritto. Consideriamo una coppia di agenti differenti *(i,j)*, e definiamo la mappa:

$$T_{i,j}(V) := (V_1, ..., V_i^*, ..., V_j^*, ..., V_N)$$

se al tempo t soltanto i due agenti i e j effettuano il partizionamento a coppie, allora $V(t+1) = T_{i,j}(V(t))$. Sia ora la funzione polidroma

$$T(V) = \{T_{i,j}(V) \mid (i,j) \in \{1, ..., N\}^2,\ i \neq j\}$$

allora l'uguaglianza appena vista può essere scritta come $V(t+1) \in T(V(t))$.

Remark 9 Una funzione polidroma $f : X \to P(Y)$ associa ad ogni elemento di X una sottoinsieme non vuoto di Y.

Proposition 10 (esistenza di una sequenza temporale) Dato un inseme di N agenti che implementano l'algoritmo gossip di partizionamento a coppie, esiste sicuramente una sequenza crescente di istanti temporali (t_k), tale che $V(t_k+1) = T_{i,j}(V(t_k))$ per una coppia di agenti (i,j).

La dimostrazione di tale teorema si può trovare in [16], e deriva direttamente dal fatto che le coppie di agenti vengono scelte in maniera casuale. L'evoluzione di un algoritmo gossip che utilizza il partitioning a coppie, può essere descritto come un processo discreto nel tempo, in cui, assumendo $V(t_k+1) := V(k+1)$ e $V(t_k) := V(k)$, si ha

$$V(k+1) \in T(V(k))$$

definendo con I_k il vettore di informazione che descrive completamente lo stato dell'algoritmo gossip successivamente all'iterazione k-esima del partitioning a coppie, è verificata la seguente proposizione:

Proposition 11 (probabilità di comunicazione) Data un insieme di N agenti che implementano l'algoritmo gossip di partizionamento a coppie, allora esiste un numero reale $p \in (0,1)$, tale che per ogni $k \in \mathbb{Z}_+$e per ogni coppia di agenti (i,j):

$$P[V(k+1) = T_{i,j}(V(k)) \mid I_k] \geq p \tag{19}$$

questa proposizione è dimostrata in [16] e può essere riformulata definendo la coppia che comunica all'istante k come $\sigma(k) = \{(i,k) \in \{1, ..., N\}^2, i \neq j\}$, allora la sequenza di coppie di agenti agli istanti (t_k), può essere vista come una realizzazione di $\sigma(k)$ e la disuguaglianza (19) è equivalente a

$$P[\sigma(k+1) = (i,j) \mid \sigma(k)] \geq p \tag{20}$$

il prossimo passo è quello di dimostrare che la funzione costo decresce ogni volta che l'algoritmo di partitioning modifica le regioni assegnate ai vari agenti [16].

Proposition 12 (decrescenza del costo) Siano V la partizione di Q e $V^+ \in T(V)$. Se $V^+ \neq V$, allora $H_{centroid}(V^+) < H_{centroid}(V)$.

Proof Assumiamo che *(i,j)* sia la coppia di agenti che sta eseguendo il partitioning a coppie. Allora

$$H_{centroid}(V^+) - H_{centroid}(V) =$$
$$H_{one}(V_i^+, C(V_i^+)) + H_{one}(V_j^+, C(V_j^+))$$
$$-H_{one}(V_i, C(V_i)) - H_{one}(V_j, C(V_j))$$

per come abbiamo definito l'algoritmo di partitioning, $V_i^+ \neq V_i$ e $V_j^+ \neq V_j$, quindi

$$H_{one}(V_i^+, C(V_i^+)) + H_{one}(V_j^+, C(V_j^+)) < H_{one}(V_i, c_i) + H_{one}(V_j, c_j).$$

Le tre proposizioni viste sono necessarie per poter applicare il seguente teorema, la cui dimostrazione può essere trovata in [26].

Theorem 13 (scambi casuali implicano convergenza) Sia (X,d) uno spazio metrico. Data una collezione di mappe $T_1, ..., T_m : X \to X$, una funzione polidroma $T(x) = \{ T_1(x), ..., T_m(x)\}$ ed un processo stocastico $\sigma : \mathbb{Z}_+ \to \{1, ..., m\}$, consideriamo una evoluzione di T, $\{x_n\}_{n\in\mathbb{Z}_+}$tale che $x_{n+1} = T_{\sigma(n)}(x_n)$ e siano verificate:

(a) esiste un insieme $W \subseteq X$ invariante e strettamente. positivo (strongly positively invariant set) per T;

(b) esiste una funzione $U : W \to \mathbb{R}$ tale che $U(w') < U(w)$, per ogni $w \in W$ e $w' \in T(w) \setminus \{w\}$;

(c) esiste $p \in (0, 1)$ e $k \in \mathbb{N}$, tale che, per ogni $i \in \{1, ..., m\}$ e $n \in \mathbb{Z}_+$, esiste un indice $h \in \{1, ..., k\}$ tale che $P[\sigma(n + h) = i \mid \sigma(n), ..., \sigma(1)] \geq p$.

Per $i \in \{1, ..., m\}$, sia F_i l'insieme dei punti fissi di T_i in W, i.e., $F_i = \{w \in W \mid T_i(w) = w\}$. Se $x_0 \in W$, allore l'evoluzione $\{x_n\}_{n\in\mathbb{Z}_+}$ converge in un numero finito di passi ad un elemento dell'insieme $F_1 \cap ... \cap F_m$, cioè esiste sicuramente un $\tau \in \mathbb{N}$ tale che, per qualche $\bar{x} \in F_1 \cap ... \cap F_m$, $x_n = \bar{x}$ per $n \geq \tau$.

Nel caso in esame, le mappe T_i utilizzate nel teorema, sono proprio le mappe $T_{i,j}(V(t)) = V(t + 1)$ definite in precedenza, quindi la successione $x_{n+1} = T_{\sigma(n)}(x_n)$ corrisponde alla successione di partizioni ottenute tramite l'algoritmo di partitioning a coppie. Il seguente teorema sfrutta il risultato appena descritto per dimostrare che effettivamente l'algoritmo gossip a coppie converge. Anche in questo caso è stato necessario andare a modificare la dimostrazione presente in [16], soprattutto per quanto riguarda la seconda parte del teorema, in modo tale da ottenere il risultato voluta anche senza associare un grafo alla mappa.

Theorem 14 (Convergenza Pairwise gossip Partitioning) Data una rete di N agenti con proprietà 1), 2) e 3), se questi implementano un algoritmo gossip di partizionamento secondo l'algoritmo Pairwise Partitioning, selezionando ad ogni iterazione due agenti adiacenti in maniera casuale, allora:

(i) Ogni cella di $V_i(t)$ è raggiungibile da una qualunque altra cella di $V_i(t)$;

(ii) V(t) converge in un numero finito di passi ad una partizione a coppie ottima.

Proof ***(punto (i))*** Verifichiamo come prima cosa che $V(t + 1)$ sia effettivamente una partizione di Q. Dalla definizione dell'algoritmo $V_i(t+1) \cup V_j(t+1) = V_i(t) \cup V_j(t)$, inotre, poichè $a \in V_i(t + 1)$ e $b \in V_j(t + 1)$, $V_i(t + 1) \neq \emptyset$ e $V_j(t + 1) \neq \emptyset$. Sono, quindi, verificate le proprietà 1) e 3) di partizione. La proprietà 2), invece, deriva direttamente dal fatto che le due partizioni $V_i(t + 1)$ e $V_j(t + 1)$ sono due sottoinsiemi disgiunti di $V_i(t) \cup V_j(t)$. Rimane ora da dimostrare che ogni punto delle partizioni create, è raggiungibile da ogni altra cella della partizione stessa. Supponiamo per assurdo che non sia così, cioè che il più breve percorso che congiunge due celle x e c_i di $V_i(t) \cup V_j(t)$, contenga una cella di V_j^*, che indichiamo con la lettera m.

Si ha quindi che $d_{V_i(t)\cup V_j(t)}(m, c_j) < d_{V_i(t)\cup V_j(t)}(m, c_i)$, allora:

$$\begin{aligned} &d_{V_i(t)\cup V_j(t)}(x, c_j) \leq \\ &d_{V_i(t)\cup V_j(t)}(m, c_j) + d_{V_i(t)\cup V_j(t)}(x, m) < \\ &< d_{V_i(t)\cup V_j(t)}(m, c_i) + d_{V_i(t)\cup V_j(t)}(x, m) = \\ &d_{V_i(t)\cup V_j(t)}(m, c_i) \end{aligned} \tag{21}$$

ciò è una contraddizione per $x \in V_i^*$, in maniera analoga si può ragionare per V_j^*.

(punto (ii)) La partizione di Q è un insieme invariante e strettamente positivo, quindi il punto *(a)* del teorema (13) è verificato. Inoltre, se si sceglie come funzione U la $H_{centroid}$ anche il secondo punto è verificato. Infine la disuguaglianza (20) è equivalente a punto *(c)*. Siamo quindi nelle condizioni di applicare il teorema (13), secondo il quale si ha la convergenza in un numero finito di passi di *V(t)* ad un elemento dell'intersezione dei punti di equilibrio delle mappe $T_{i,j}$. Tale punto di equilibrio corrisponde a $T_{i,j}(V) = V$, per come è stato definito l'algoritmo di partitioning, si arriva a questa situazione solo quando il termine $H_{one}(C(V_i), V_i) + H_{one}(C(V_j), V_j)$ è minimo. Se si converge all'intersezione i punti di equilibrio corrispondenti a tutte le possibili coppie di agenti adiacenti, significa che la somma di questi due costi è minima per ogni *(i,j)*, quindi si è arrivati ad un partizionamento ottimo a coppie. ♢

1.4 Implementazione e Simulazioni

Nell'implementazione del partitioning si è partiti da una disposizione completamente casuale degli agenti all'interno della mappa, con una conseguente suddivisione casuale dello spazio a disposizione. Tramite delle funzioni appositamente create si determinano i centroidi di ogni regione, e si determinano quali sono gli agenti adiacenti, che quindi possono comunicare tra loro.

A questo punto è possibili applicare l'algoritmo di partitioning formalizzato nella sezione 1.2.1, che deve, però, essere leggermente modificato. Come anticipato, infatti, non viene fatto alcun controllo sulla possibilità che gli agenti si scambino le proprie aree di competenza. In generale

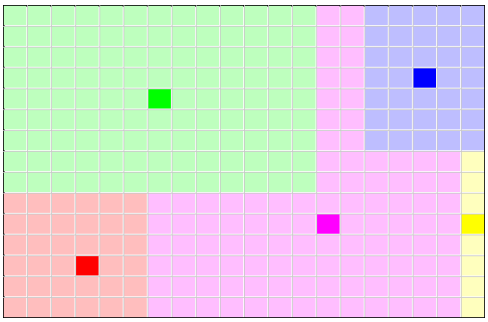

Figura 2: Suddivizione iniziale della mappa.

questo potrebbe non essere un problema, ma, nel particolare caso preso in esame, si corre il rischio di perdere tempo per effettuare inutili spostamenti. Per ovviare al problema è stato sufficiente imporre che la regione assegnata ad un agente dopo una fase di partitioning sia quella contenente la posizione iniziale dell'agente stesso. Dalle simulazioni fatte è possibile notare che, ad ogni iterazione, due agenti suddividono in maniera ottima lo spazio costituito dall'unione delle loro regioni, ottenendo in breve tempo il partizionamento ottimo dell'intera mappa.

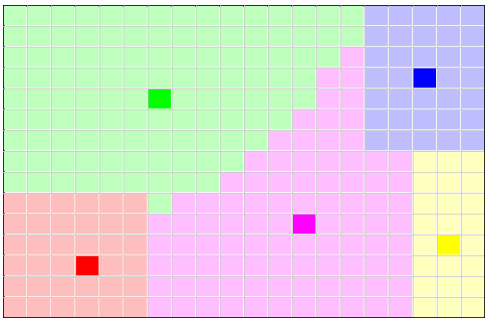

Figura 3: Terza iterazione.

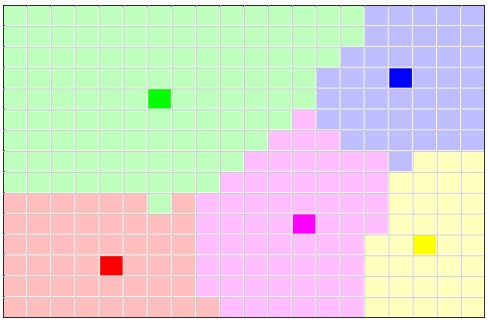

Figura 4: Sesta iterazione.

Le aree assegnate ad ogni agente, non sono essattamente uguali. Questo è dovuto al fatto che, nel caso discretizzato, le celle corrispondenti al confine tra le regioni non vengono sempre assegnate all'agente corretto. Se si suddivide la mappa con un numero maggiore di celle, si ottiene una griglia più fitta, con celle più piccole e quindi con errori minori (vedi figura 6).

Si rappresenta, infine, l'andamento dell'area delle regioni assegnate ad ogni agente. Queste sono inizialmente molto diverse, ma, ad ogni iterazione, convergono a dei valori che sono più vicini tra loro tanto maggiore è il numero di celle considerate. Dal momento che si è utilizzato un algoritmo gossip simmetrico, convergono ad un valore vicino alla media delle aree iniziali.

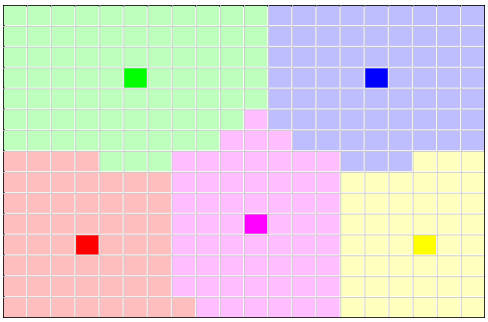

Figura 5: Ultima iterazione.

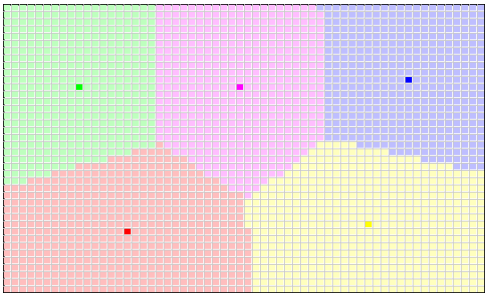

Figura 6: Partitioning con molte celle.

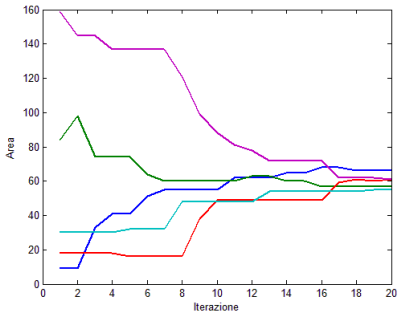

Figura 7: Convergenza delle aree delle regioni al valore medio delle aree iniziali (60 celle).

Prima di poter iniziare la stima del campo di interesse, è necessario che gli agenti compiano delle misure nella mappa Q spostandosi al suo interno. Si sono quindi inseriti gli indici delle celle di ogni agente all'interno di alcuni vettori, ed è stato creato un algoritmo concettualmente simile all'ordinamento *BubbleSort* che ordini le celle in maniera tale da far compiere agli agenti un percorso come quello rappresentato in figura 8.

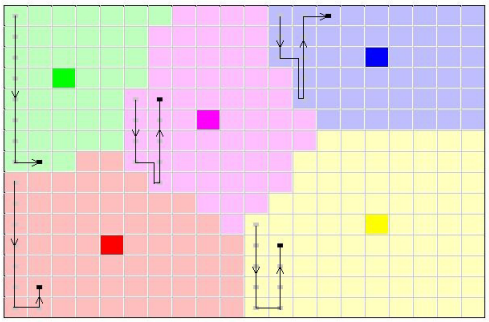

Figura 8: Strategia esplorativa adottata dagli agenti.

2 Stima del Campo di Interesse

2.1 Utilizzo di Algoritmi di Consensus per la Stima Distribuita

I problemi di stima distribuita si trovano in tutte quelle applicazioni in cui è presente un grande numero di dispositivi che possono comunicare e cooperare per raggiungere uno scopo comune. Questo tipo di sistemi possono essere estremamente efficienti in applicazioni di monitoraggio o di controllo in ambienti di grandi dimensioni, dove si adottano spesso soluzioni di tipo centralizzato, che comportano la necessita di sensori e attuatori molto affidabili e, quindi, costosi. Nei sistemi distribuiti, invece, si sfrutta la capacità di alcuni dispositivi intelligenti di cooperare e prendere decisioni in maniera autonoma, senza la necessità di supervisioni. I vantaggi principali stanno nel fatto che, si utilizzano componenti di piccole dimensioni, poco costosi ed in grado di gestire autonomamente eventuali malfunzionamenti. I principali sistemi di questo tipo sono le Wireless Sensor Networks (WSNs), le Smart Camera Networks (SCNs), le Smart Power Grids (SPGs) e le Robot Sensor Networks (RSNs). Queste ultime, come anticipato nell'introduzione, hanno svariate applicazioni, ad esempio, il patrolling, la copertura di ambienti estesi, la misura distribuita, il map-building distribuito e la navigazione autonoma. Una possibile applicazione del sistema considerato in questo progetto potrebbe essere la ricerca, da parte di alcuni agenti robotici, di persone disperse in ambienti ostili, a seguito di disastri ambientali o incidenti. Ogni agente dovrebbe essere dotato di sensori di temperatura o sensori ad infrarossi, che gli conferirebbero la capacità di rilevare il calore emanato dal corpo umano. I vari agenti, quindi, si dovrebbero disporre in maniera ottima all'interno dell'ambiente, dovrebbero effettuare le misurazioni nella propria regione di interesse, e, tramite un algoritmo di stima distribuita, dovrebbero determinare una mappatura completa delle misure nell'intera mappa. In questo modo, è possibile ottenere una mappa preliminare dei luoghi in cui concentrare maggiormente le ricerche, con un ovvio vantaggio dal punto di vista del tempo necessario alle ricerche e dal punto di vista della sicurezza del personale addetto. Per risolvere questo tipo di problemi è possibile utilizzare degli algoritmi di consenso lineari, che si basano fondamentalmente sul calcolare medie delle quantità locali presenti nei vari dispositivi e che, nel caso particolare della comunicazione gossip, sono già stati descritti brevemente nella sezione 1.2.1.

2.1.1 Minimi Quadrati

Supponiamo di avere un modello di misura lineare del tipo:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} f(x_1, \theta) \\ \vdots \\ f(x_N, \theta) \end{bmatrix} + \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \mathbf{f}(\theta) + \mathbf{w}$$

in cui y_i costituisce la misura rilevata dall'agente in questione nella posizione x_i, $\mathbf{f}(\theta)$ è la funzione vera che descrive il campo misurato in quel punto dal sensore dell'agente, w_i è il rumore di misura e N_Q è il numero di misure a disposizione dell'agente. Il metodo dei minimi quadrati permette di determinare in maniera semplice una stima di $\mathbf{f}$

$$\hat{\mathbf{y}}(\theta) = \mathbf{f}(\theta).$$

Per poter procedere è necessario fare un'ipotesi fondamentale, cioè che la funzione vera che descrive il nostro campo di interesse, possa essere approssimata da una combinazione lineare di Radial Basis Function (RBF), cioè di particolari funzioni reali, il cui valore dipende esclusivamente dalla distanza dal proprio centro (e.g. funzione gaussiana, vedi figura 9).

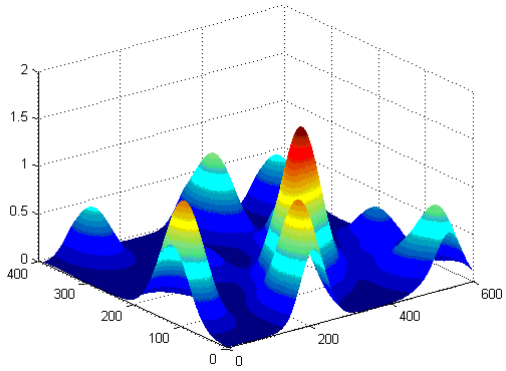

Figura 9: Funzione vera come somma di RBF (nell'esempio 10 RBF).

Sotto questa ipotesi è possibile rappresentare la funzione nel modo seguente. Sia

$$s(x_i) := \begin{bmatrix} \phi_1(x_i) & \cdots & \phi_p(x_i) \end{bmatrix}$$

Allora

$$f(x_i, \theta) = s(x_i)\theta = \sum_{k=1}^{p} \phi_k(x_i)\theta_k$$

dove p corrisponde al numero di funzioni radiali $\phi_k(x)$ utilizzate nell'approssimazione e il vettore $s(x)$ contiene i valori assunti da tali funzioni nel punto x. Il modello lineare diviene

$$y_i = s(x_i)\theta + w_i$$

In figura 10 è rappresentato un esempio del campo misurato dai vari agenti, che corrisponde alla funzione vera rappresentata in figura 9, a cui viene sommato un rumore bianco.

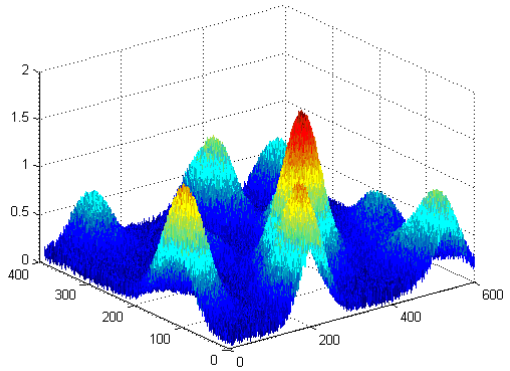

Figura 10: Esempio di campo misurato dai sensori ($\sigma_w = 0.2$).

Definiamo

$$\mathbf{C} = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_p(x_1) \\ \phi_1(x_2) & \cdots & \phi_p(x_2) \\ \vdots & \ddots & \vdots \\ \phi_1(x_{N_Q}) & \cdots & \phi_p(x_{N_Q}) \end{bmatrix} = \begin{bmatrix} s(x_1) \\ s(x_2) \\ \vdots \\ s(x_{N_Q}) \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y(x_1) \\ \vdots \\ y(x_{N_Q}) \end{bmatrix} = \mathbf{C}\theta + \mathbf{w} \qquad N_Q = \dim Q$$

per semplicità di notazione, supponiamo di rappresentare l'insieme di tutte le misure nella mappa Q tramite il vettore $\mathbf{y}$ e i valori assunti dalle RBF all'interno di Q tramite la matrice $\mathbf{C}$. Allora la soluzione al problema di minimizzazione della cifra di merito è, come noto :

$$\begin{aligned} \hat{\theta}(\mathbf{y}) &= \left[\mathbf{C}^T\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{y} \qquad (22) \\ &= \left[\frac{1}{N_Q}\sum_{i=1}^{N_Q} s(x_i)^T s(x_i)\right]^{-1} \frac{1}{N_Q}\sum_{i=1}^{N_Q} s(x_i) y_i \end{aligned}$$

in appendice è presente una breve dimostrazione. In questo modo, è possibile ottenere una stima del campo di interesse moltiplicando C per lo stimatore:

$$\hat{y}_i = s(x_i)\hat{\theta} = \sum_{k=1}^{p} \phi_k(x_i)\hat{\theta}_k \qquad (23)$$

in forma vettoriale:

$$\hat{\mathbf{y}} = \mathbf{C}\hat{\theta} \qquad (24)$$

questa soluzione, però, prevede di utilizzare esclusivamente le osservazioni del singolo agente, supponendo che questo possieda le misure del campo in tutta la mappa Q. Utilizzandolo con più agenti, si otterrebbe un insieme di stime locali del campo di interesse che ne descrivono l'andamento in maniera parziale e poco esplicativa (vedi figura 11). É evidente che conviene adottare un approccio di tipo distribuito, in modo da ottenere una stima del campo che lo descriva globalmente in tutto l'ambiente in cui vengono fatte misurazioni. La soluzione adottata è una stima ai minimi quadrati distribuita ottenuta tramite un algoritmo gossip simmetrico.

Sia N il numero di agenti utilizzati e indichiamo con Q_h la regione di interesse dell'agente h-esimo, per prima cosa ognuno di essi calcola il proprio *vettore di informazione*, cioè le due componenti

$$\left[\ {\mathbf{C}_h}^T\mathbf{C}_h \quad {\mathbf{C}_h}^T\mathbf{y}\ \right] \qquad h = 1, ..., N$$

$$\mathbf{C}_h = \begin{bmatrix} \phi_1(x_{1_h}) & \cdots & \phi_p(x_{1_h}) \\ \phi_1(x_{2_h}) & \cdots & \phi_p(x_{2_h}) \\ \vdots & \ddots & \vdots \\ \phi_1(N_{Q_h}) & \cdots & \phi_p(x_{N_{Q_h}}) \end{bmatrix} \qquad N_{Q_h} = \dim Q_h$$

dove le $x_{k_h} \in Q_h$ sono i punti in cui l'agente h-esimo ha delle misure, ed è possibile ottenere una stima locale del campo.

A questo punto ha inizio la stima distribuita vera e propria. Ogni agente trasmette una richiesta di comunicazione in maniera Broadcast a tutti gli agenti vicini e, seguendo un certo protocollo di comunicazione, inizia a scambiare informazioni con uno soltanto di questi (vedi figura 12).

Una volta ricevuto il vettore di informazione, ogni agente compie una operazione di media con il proprio vettore, ottenendo, all'iterazione i-esima:

$${\mathbf{C}_h}^T\mathbf{C}_h(i) = \frac{{\mathbf{C}_h}^T\mathbf{C}_h(i-1) + {\mathbf{C}_k}^T\mathbf{C}_k(i-1)}{2} \qquad (25)$$

$${\mathbf{C}_h}^T\mathbf{y}(i) = \frac{{\mathbf{C}_h}^T\mathbf{y}(i-1) + {\mathbf{C}_k}^T\mathbf{y}(i-1)}{2} \qquad (26)$$

entrambi gli agenti in questione avranno lo stesso vettore di informazione dopo ogni iterazione. Questa operazione di media viene effettuata tra i vari agenti fino a quando tutti gli agenti hanno lo stesso vettore di informazione. In fase implementativa si è costruito un ciclo in cui, ad ogni iterazione, si simula la comunicazione gossip determinando la lista degli agenti adiacenti e se ne selezionano due vicini in maniera totalmente casuale. Per verificare se l'aggiornamento del vettore di informazione di ogni agente non comporta nessun cambiamento e si è arrivati alla convergenza, si è deciso di confrontare il vettore di informazione all'istante corrente con quello precedente all'operazione di media, terminando il ciclo solo quando non si riscontra alcuna variazione per un numero di cicli prefissato e sufficientemente elevato. Una volta arrivati alla convergenza, si ha che tutti gli agenti determinano esattamente la stessa stima del campo di interesse e il numero di iterazioni necessarie ad avere la convergenza è dello stesso ordine di grandezza del numero di celle in cui è stata divisa la mappa. La stima ottenuta tramite il metodo dei minimi quadrati distribuito è rappresentata in figura 13.

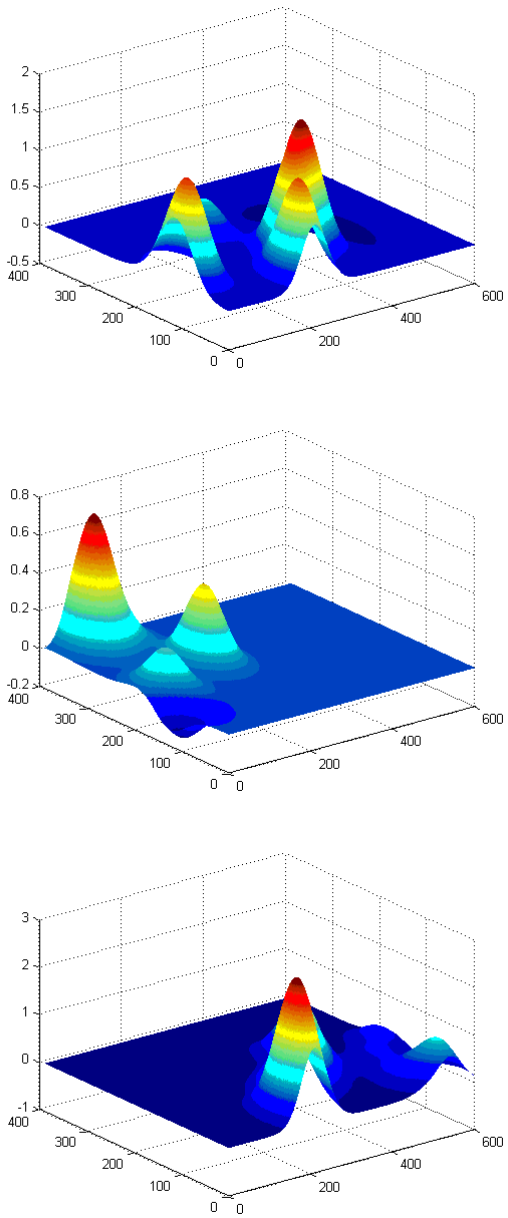

Figura 11: Stima locale del campo di interesse. I tre grafici corrispondono al campo stimato localmente da tra diversi agenti sulla base delle proprie misure.

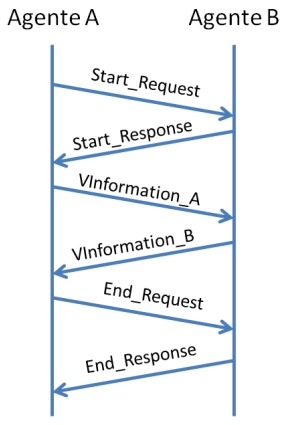

Figura 12: Comunicazione gossip.

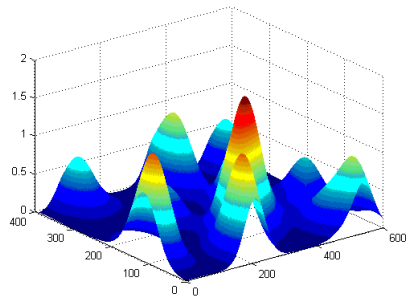

Figura 13: Stima MQ Distribuita.

Di seguito vengono presentati in formato di pseudo codice i due algoritmi implementati per ottenere il calcolo del vettore di informazione e la stima distribuita appena descritta. Oltre alle variabili già definite, viene utilizzata la matrice *Ad*, in cui, ad ogni riga associata ad un agente, sono presenti tutti gli agenti ad esso adiacenti. Per simulare la comunicazione gossip tra due agenti adiacenti si è scelto di selezionarne due dalla matrice in maniera casuale.

---

function InformationVector
  $C^TY_{temp} \leftarrow 0$ // variabile temporanea per
    // il calcolo di C^TY
  $C^TC_{temp} \leftarrow 0$ // variabile temporanea per
    // il calcolo di C^TC
  for all agents h in Q **do**
    for all $(x_i, y_i) \in Q_h$ **do**
      $C^TY_{temp} \leftarrow C^TY_{temp} + s(x_i)^T y_i$
      $C^TC_{temp} \leftarrow C^TC_{temp} + s(x_i)^T s(x_i)$
    $C^TY_h \leftarrow C^TY_{temp}$
    $C^TC_h \leftarrow C^TC_{temp}$
  return (C^TY, C^TC)

function DistEstimation(Ad, C^TC, C^Ty, γ)
  $n \leftarrow 0$
  while $n < 100$ **do**
    $h \leftarrow$ agente nella mappa Q
    $k \leftarrow$ agente in Ad[h]
    $temp_h \leftarrow C^Ty_h$ //variabile temporanea
    $temp_k \leftarrow C^Ty_k$ //variabile temporanea
    $C^TC_h, C^TC_k \leftarrow (C^TC_h + C^TC_k)/2$
    $C^TY_h, C^TY_h \leftarrow (C^Ty_h + C^Ty_k)/2$
    if $temp_h = C^Ty_h$ **then**
      $n \leftarrow n + 1$
    else
      $n \leftarrow 0$
    if $temp_k = C^Ty_k$ **then**
      $n \leftarrow n + 1$
    else
      $n \leftarrow 0$
  $\hat{\theta} \leftarrow (C^TC + \gamma I)^{-1} C^TY$
  $\hat{y} \leftarrow s\hat{\theta}$
  return $\hat{y}$

2.2 Regressione Lineare e Regolarizzazione di Tikhonov

L'analisi della regressione si occupa di determinare la relazione esistente tra una variabile y continua (variabile dipendente), e uno o più variabili indipendenti. Consideriamo cioè una relazione:

$$y = f(\phi_1, ..., \phi_k) + \epsilon$$

che rappresenta il legame tra la variabile dipendete e quelle indipendenti, a cui va sommato un termine di errore che non può essere predetto. Il modello più utilizzato è quello lineare

$$y = \beta_0 + \beta_1\phi_1 + ... + \beta_k\phi_k + \epsilon$$

dove β_0 è detto termine noto, mentre i $\beta_1, ..., \beta_k$ sono detti coefficienti di regressione, e costituiscono i parametri da stimare sulla base delle osservazioni a disposizione. Il metodo dei minimi quadrati permette di determinare la stima di questi parametri tramite l'equazione vista nella sezione 2.1.1

$$\hat{\beta}(\mathbf{y}) = \left[\mathbf{C}^T\mathbf{C}\right]^{-1}\mathbf{C}^T\mathbf{y} \tag{27}$$

una delle ipotesi che bisogna fare per poter calcolare tale stima è che la matrice delle variabili indipendenti $\mathbf{C}$ abbia rango pieno, in modo tale che $\mathbf{C}^T\mathbf{C}$ risulti essere non singolare, e quindi invertibile. Nelle fasi iniziali della stima, quando i dati a disposizione sono pochi, si può creare una situazione di quasi singolarità, che porta ad avere degli elementi sulla diagonale di $[\mathbf{C}^T\mathbf{C}]^{-1}$ molto elevati e quindi una stima poco attendibile dei coefficienti di regressione. Questo problema si è spesso presentato nell'implementazione del metodo dei minimi quadrati distribuiti, in alcuni casi si è addirittura verificato un overflow nell'inversione della matrice, che ovviamente ha portato all'interruzione del calcolo della stima. I metodi più utilizzati per risolvere la quasi singolarità sono:

1. l'aggiunta di nuove osservazioni che rendano la matrice C a rango pieno;
2. l'esclusione dal modello delle variabili correlate, cioè delle variabili per cui la stima della varianza del coefficiente di regressione associato è elevata;
3. l'uso della principal component regression (PCR): si sostituiscono le variabili indipendenti con altre variabili ortogonali, ottenute estraendone le componenti principali. Si veda [27];
4. l'uso della *regolarizzazione di Tikhonov*.

La soluzione implementata è la *Ridge Regression*, un particolare algoritmo derivato direttamente dalla regolarizzazione di Tikhonov, che si basa sul pesare i coefficienti di regolarizzazione, minimizzando la somma quadratica

$$\hat{\beta}_{TR} = \arg\min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{k}\phi_j\beta_j)^2 + \gamma\sum_{j=1}^{k}{\beta_j}^2\right\} \tag{28}$$

dove $\gamma \leq 0$ determina la compressione dei vari parametri [19]. La funzione costo può essere riscritta in forma vettoriale

$$R(\gamma) = (\mathbf{y} - \mathbf{C}\beta)^T(\mathbf{y} - \mathbf{C}\beta) + \gamma\beta^T\beta \tag{29}$$

ed è facile verificare che il minimo è raggiunto per

$$\hat{\beta}_{TR} = (\mathbf{C}^T\mathbf{C} + \gamma\mathbf{I}_k)^{-1}\mathbf{C}^T\mathbf{y}. \tag{30}$$

In questo modo si va a sommare un termine costante alla diagonale di $\mathbf{C}^T\mathbf{C}$, assicurando che il problema sia non singolare, anche quando la matrice non ha rango pieno. Per determinare la stima ai minimi quadrati descritta nella sezione 2.1.1, si è quindi deciso di adottare proprio questo procedimento, sommando un termine costante al prodotto $\mathbf{C}^T\mathbf{C}$. Nelle fasi iniziali si è diciso di scegliere un valore di γ piccolo, ma comunque sufficientemente elevato da eliminare il problema della quasi singolarità. Il passo successivo è quello di determinare il valore di γ che risolva il problema descritto e che permetta di ottenere il modello con errore di predizione minimo. Di seguito vengono descritti i metodi adottati per la determinazione di questo valore, prendendo in considerazione sia algoritmi di tipo centralizzato sia algoritmi di tipo distribuito.

2.2.1 Parametro di Regolarizzazione in Contesto Centralizzato

L'approccio più semplice a questo problema è ovviamente quello centralizzato, in cui si suppone che esista un unico agente che compie le misure in tutta la mappa o che vi sia un'unità centrale che conosca le misure dei vari agenti e che sia, quindi, in grado di determinarne una stima globale. Nel caso preso in esame si è supposto che tutti gli agenti possano prendere misure nella propria regione, e che vi sia uno di questi agenti (*Master*) che abbia una unità di calcolo e di memoria in grado di immagazzinare tutte le misure prese dagli altri agenti (*Slave*) e di elaborare la stima del campo di interesse. Per prima cosa il Master deve calcolare il vettore di informazione relativo a tutte le misure, successivamente, tramite le solite formule (22) e (23), determina la stima del campo di interessa nell'intera mappa Q. La scelta più naturale, per quanto riguarda il γ ottimo, ricade sul valore che minimizza l'errore di stima. Si è quindi deciso di calcolare l'RMS di quest'ultimo per un insieme prefissato di γ, e di selezionare quello, tra questi, che porta ad avere l'errore più basso. Dal momento che l'insieme dei γ sperimentati non è continuo, non è possibile convergere esattamente al γ ottimo, ma, scegliendo dei valori sufficientemente vicini tra loro, è possibile comunque ottenere un risultato soddisfacente. Sia $\Gamma = (\gamma_1, ..., \gamma_q)$ l'insieme dei gamma scelto, il valore ottimo sarà:

$$\gamma_{ott} = \arg\min_{\gamma\in\Gamma}\sqrt{\frac{1}{N_Q}\sum_{i=1}^{N_Q}(\hat{y}_i - y_i)^2} \tag{31}$$

dove $\hat{y}_i$ e y_i sono rispettivamente la stima del campo di interesse ed il valore misurato dello stesso nel punto $x_i \in Q$. Nell'implementazione dell'algoritmo si è scelto, per prima cosa, di utilizzare un insieme Γ piuttosto esteso, in modo da capire in maniera quantitativa l'andamento dell'RMS al variare di γ. Successivamente si è ristretto l'insieme ad un range di valori più significativo e che permetta di ottenere un valore ottimo più preciso.

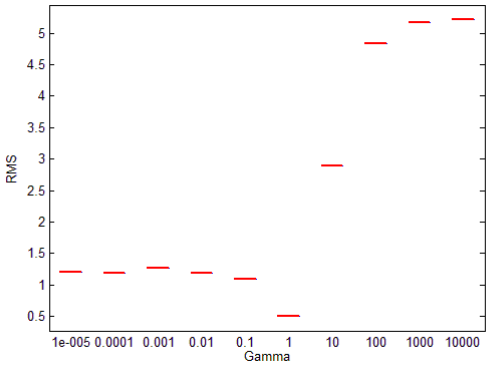

Figura 14: γ_{ott} centralizzato con range esteso.

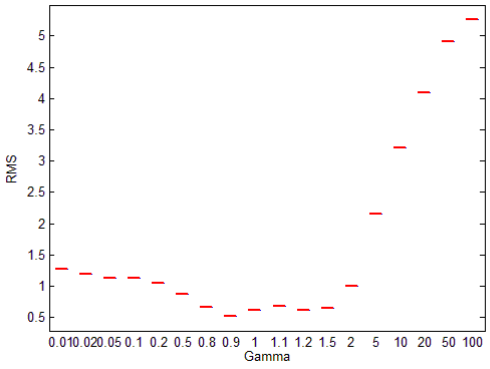

Figura 15: γ_{ott} centralizzato con range ridotto.

Come si può notare dalla figura 15, l'andamento dell'RMS non è molto regolare. Ciò avviene perchè si esegue una sola simulazione per ogni valore di γ, quindi il risultato ottenuto è decisamente influenzato dalla particolare realizzazione del rumore sulle misure. Conviene quindi effettuare un numero di simulazioni sufficientemente elevato (vengono effettuate 100 simulazioni per ogni γ) e successivamente fare una media tra gli RMS ottenuti.

2.2.2 Metodo della Cross Validation

La scelta del parametro di regolarizzazione descritta nella sezione precedente, è stata ottenuta mantenendo sempre le stesse funzioni radiali e utilizzando le misure relative a funzioni reali diverse, corrispondenti cioè a parametri θ differenti. Si è scelto di agire in questo modo perché, considerando una sola stima del campo, la scelta del γ risulta essere molto influenzata dalla singola realizzazione del rumore di misura. Questa soluzione, però, non è molto adat-

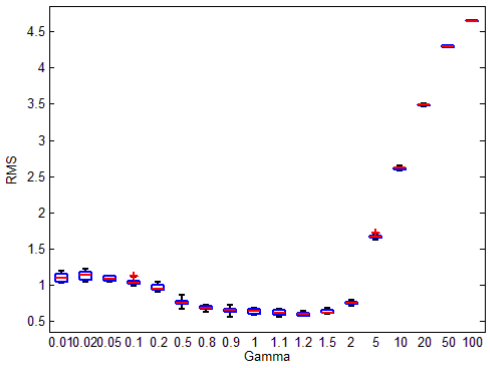

Figura 16: BoxPlot dell'RMS ottenuto in maniera centralizzata.

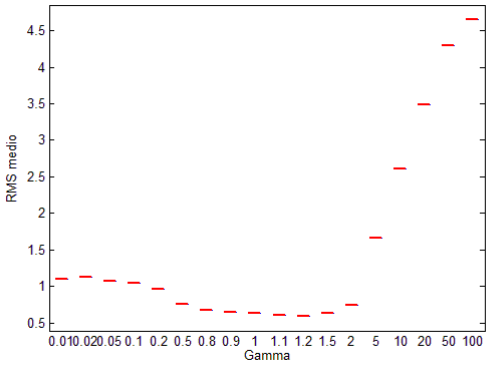

Figura 17: Valori medi dell'RMS ottenuto in maniera centralizzata.

ta al tipo ti problematiche che si cerca di risolvere in questo progetto, perchè, in generale, il campo di interesse che si vuole stimare è solamente uno, e non si ha la possibilità di confrontare un grande numero di simulazioni ottenendo un comportamento medio del sistema. Altro problema che si presenta è che si trattano in maniera troppo ottimistica i dati, infatti si cerca di minimizzare l'RMS utilizzando tutte le misure a disposizione, e si corre il rischio di trovare un modello del sistema che approssima molto bene i dati in questione, ma non tiene conto del fatto che, mano a mano che arrivano nuove misure, la stima del campo si modifica e con essa anche il γ ottimo. Per risolvere questi problemi si utilizza una particolare implementazione del metodo della CV, che permette di determinare il γ in grado di massimizzare le capacità predittive del modello e di effettuare più simulazioni sui dati, in modo da ottenere un comportamento medio del sistema.

L'idea di base è quella di suddividere i dati a disposizione, detti *data set*, in due parti, la prima, detta *training set,* viene utilizzata per identificare il modello del campo di interesse, mentre la seconda, detta *validation set*, viene utilizzata per verificare la bontà della stima effettuata. Se il validation set è troppo piccolo, però, la stima delle performance ottenuta potrebbe avere una varianza elevata. Per ridurre questo problema si può utilizzare il metodo detto *k-fold cross-validation*, secondo il quale si suddividono i dati in k sottoinsiemi di eguale dimensione; la validazione è fatta su uno di questi sottoinsiemi, mentre l'identificazione viene fatta sui restanti k-1 insiemi, ripetendo questo procedimento per tutte e k le possibili combinazioni di validation set e training set. In pratica, una volta scelti i parametri di regolarizzazione che si vogliono confrontare, per ognuno di essi se effettuano le seguenti operazioni:

1. Effettuo una misurazione del campo, ottenendo la funzione vera più il rumore di misura;
2. Suddivido il data set in k parti uguali, ottenendo quindi k combinazioni differenti di training set (S_t) e validation set (S_v):

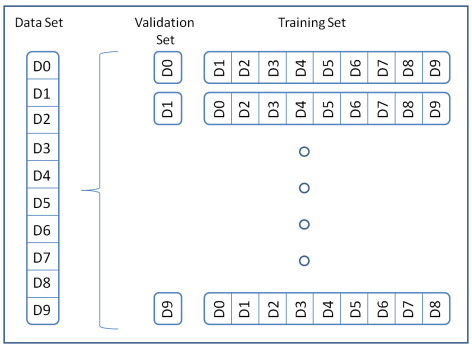

Figura 18: Suddivisione dei set nella CV.

Siano N_{v_i} e N_{t_i} il numero di dati rispettivamente di S_{v_i} e di S_{t_i}, mentre siano x_j e y_j rispettivamente la posizione e il valore del dato j-esimo di un validation set;

3. Per ogni coppia (S_{t_i}, S_{v_i}), determino la stima del campo $\hat{y}_i$ e calcolo l'RMS nei punti corrispondenti ai dati di validazione, che dipenderà dal particolare γ considerato:

$$RMS_i(\gamma) = \sqrt{\frac{1}{N_{v_i}} \sum_{j=1}^{N_{v_i}} (\hat{y}_i - y_j)^2} \qquad (32)$$
$$i = 1, ..., N_{t_i};$$

4. Calcolo l'RMS medio tra quelli trovati.

A questo punto, la stima del γ ottimo sarà quella a cui corrisponde il valore minimo degli RMS medi calcolati. A differenza del metodo visto nella sezione precedente, in questo caso, determino un valore di γ che minimizza l'errore rispetto ai dati di validazione, e non più rispetto ai dati di identificazione.

Ci si basa, quindi, sul principio che un modello di un fenomeno è tanto migliore quanto meglio riesce a prevedere un valore qualunque del fenomeno stesso. Quindi il parametro ottimo sarà quello per cui si realizza la migliore previsione di un dato qualunque del problema. Il vantaggio di scegliere questo tipo di soluzione, sta nel fatto che si trova una stima che non dipende esclusivamente dai dati a disposizione, ed è quindi in grado di descrivere bene il sistema anche con misure differenti da quelle utilizzate inizialmente. In figura 19 è rappresentato il risultato ottenuto applicando la CV in maniera centralizzata, cioè supponendo che vi sia un agente Master che possiede le misure del campo nell'intera mappa Q, ed è quandi in grado di applicare il metodo dull'intero set di dati a disposizione.

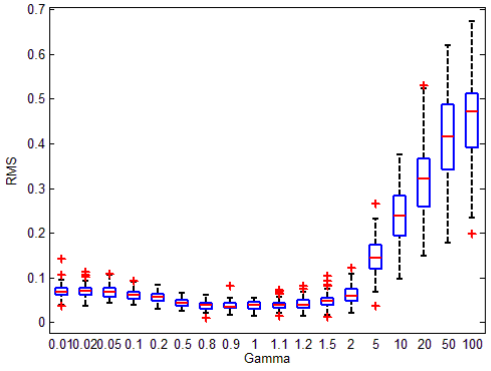

Figura 19: CV Centralizzata.

il procedimento appena descritto viene formalizzato sotto forma di pseudo codice, dove con nSet si indica il numero di set che si è deciso di utilizzare nel training set, con RMS si indica il vettore di RMS ottenuti per un dato γ e con DS, VS e TS si indicano rispettivamente il data

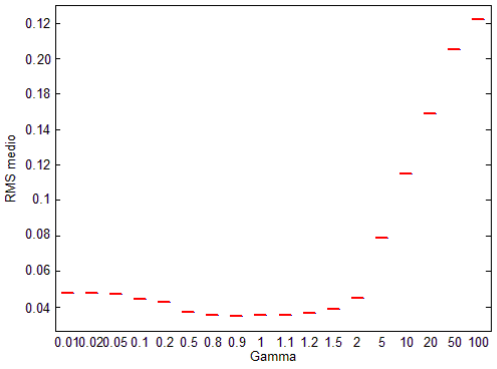

Figura 20: CV Centralizzata, valori medi.

set, il validation set e il training set. Viene, inoltre, utilizzato un metodo *createSet(nSet)*, che suddivide in maniera casuale le celle della mappa in sottoinsiemi, in modo da creare il data set.

---

function CENTRALIZEDCV(γ, nSet)
  $DS \leftarrow nSet$
  for $i = 1 \rightarrow nSet$ **do**
    $C^T y \leftarrow 0$
    $C^T y \leftarrow 0$
    $VS \leftarrow$ primo set di dataSet
    $TS \leftarrow$ restanti set di dataSet
    $DS \leftarrow$ //rotazione di DS per cambiare
      //i successivi VS e TS
    for all $(x_i, y_i) \in TS$ **do**
      $C^T Y \leftarrow C^T Y + s(x_i)^T y_i$
      $C^T C \leftarrow C^T C + s(x_i)^T s(x_i)$
    $\hat{\theta} \leftarrow (C^T C + \gamma I)^{-1} C^T Y$
    $\hat{y} \leftarrow s\theta$
    $somma \leftarrow 0$
    for all $(x_i, y_i) \in VS$ **do**
      $somma \leftarrow somma + (\hat{y}_i - y_i)^2$
    $N \leftarrow length(VS)$
    $RMS[i] \leftarrow sqr(somma \backslash N)$
  return RMS

La soluzione trovata non va ancora bene per lo scopo di questo progetto, infatti, come descritto nelle sezioni precedenti, si vuole adottare un insieme di agenti che non abbiano un sistema centralizzato e che non siano in grado di comunicare direttamente con tutti gli altri agenti. Si rende necessario modificare questo procedimento in modo da adattarlo al contesto distribuito in cui ci troviamo, introducendo quindi un termine di consenso tra gli agenti e ottenendo una stima condivisa del γ ottimo, mantenendo, però, soltanto le proprie informazioni locali.

2.2.3 Cross Validation in un Contesto Distribuito

Per determinare il γ ottimo in maniera distribuita, si è deciso di suddividere l'algoritmo in quattro parti fondamentali, alternando operazioni svolte localmente da ogni agente, con operazioni svolte in maniera distribuita con comunicazione gossip:

fase 1 (locale) ogni agente suddivide le proprie misure in *validation set* e *training set*; calcola successivamente il vettore di informazione relativo alle misure del training set;

fase 2 (distribuita) gli agenti utilizzano i vettori di informazione calcolati per ottenere una stima distribuita del campo di interesse nell'intera regione Q;

fase 3 (locale) ogni agente calcola l'RMS dell'errore di stima nei punti relativi alle misure del validation set;

fase 4 (distribuita) tramite un algoritmo di consensus gli agenti compiono la media tra i popri RMS in modo da arrivare ad un valore comune.

Vengono ora descritte in maniera più esaustiva le quattro fasi elencate. Nella prima fase gli agenti suddividono il proprio set di dati nei due sottoinsiemi citati. A differenza della CV centralizzata, in questo caso, si può creare il problema che il set in questione contenga un numero piuttosto ridotto di misure ed, utilizzando il metodo deterministico visto nella sezione 2.2.2, potrebbe non essere possibile creare il validation set e il training set. Se, ad esempio, uno degli agenti avesse una sola misura a disposizione, l'algoritmo verrebbe bloccato, poiché non sarebbe possibile suddividere in due parti il data set. Utilizzando le misure degli altri agenti per validare il modello, però, sarebbe comunque possibile portare a termine l'algoritmo. É stato quindi adottato un approccio probabilistico, in cui ogni agente sceglie con una certa probabilità p (nel caso implementato 20%) di inserire il dato nel proprio validation set, e con probabilità $1 - p$ (80%) di inserirlo nel training set. In questo modo, anche in presenza di una sola misura per ogni agente, alcuni di questi inseriranno il proprio dato nel validation set, e altri nel training set, rendendo l'algoritmo robusto e sempre funzionante. A questo punto il calcolo del vettore di informazione viene fatto in maniera analoga alla sezione 2.1.1, con l'importante differenza che si considerano soltanto le misure contenute nel training set, che indichiamo con T_h. Fissando N come numero di agenti, per ognuno di essi si ha che:

$$\left[\begin{array}{cc} \mathbf{C}_h{}^T \mathbf{C}_h & \mathbf{C}_h{}^T \mathbf{y} \end{array} \right] \quad h = 1, ..., N \tag{33}$$

$$\mathbf{C}_h = \begin{bmatrix} \phi_1(x_{1_h}) & \cdots & \phi_p(x_{1_h}) \\ \phi_1(x_{2_h}) & \cdots & \phi_p(x_{2_h}) \\ \vdots & \ddots & \vdots \\ \phi_1(x_{T_h}) & \cdots & \phi_p(x_{N_{T_h}}) \end{bmatrix} \quad N_{T_h} = \dim T_h$$

nella seconda fase si compie la stima distribuita descritta nella sezione 2.1.1, gli agenti, a due a due, compiono le media dei loro vettori di informazione, fino ad arrivare al consenso. In questo caso, però, la stima del parametro θ, e quindi del campo di interesse, si basa esclusivamente sulle misure del training set. Sia $\begin{bmatrix} \mathbf{C}^T\mathbf{C} & \mathbf{C}^T\mathbf{y} \end{bmatrix}$ il vettore a cui si converge, la stima sarà data dalle formule (41) e (24). Indichiamo ora il validation set dell'agente h-esimo con V_h. Nella terza fase, gli agenti calcolano localmente l'RMS dell'errore di predizione sui dati presenti nel validation set, in moto tale da ottenere una quantità, dipendente dal parametro di regolarizzazione, che descriva la capacità del modello di descrivere dei dati che non sono stati utilizzati in fase di identificazione. L'RMS si ottiene tramite l'equazione

$$RMS_h(\gamma) = \sqrt{\frac{1}{N_{V_h}} \sum_{j=1}^{N_{v_i}} (\hat{y}_j - y_j)^2} \tag{34}$$

con $N_{V_h} = dim V_h$.

Nell'ultima fase si applica un algoritmo gossip di consenso per ottenere un RMS dell'errore di predizione del modello, comune a tutti gli agenti. Come per la stima distribuita, gli agenti comunicano a due a due, ma, invece di scambiarsi i vettori di informazione, si scambiano gli RMS e compiono una operazione di media

$$RMS_h(i) = \frac{RMS_h(i-1) + RMS_k(i-1)}{2} \tag{35}$$

e, dopo un numero finito di iterazioni, arrivano tutti ad un unico valore. Questo algoritmo viene applicato per tutti i $\gamma \in \Gamma$, ottenendo quindi una serie di RMS. A questo punto è possibile determinare il parametro di regolarizzazione ottimo che porta ad avere un errore di predizione minimo sul validation set.

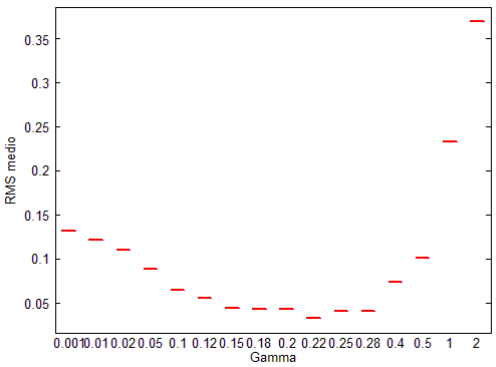

Figura 21: CV Distribuita.

lo pseudo codice dell'algoritmo è rappresentato di seguito, vengono utilizzati i simboli già definiti in precedenza nella sezione 2.2.2, con l'aggiunta dei validation set e traning set associati ad ogni agente, che vengo indicati rispettivamente con VS_h e TS_h. Come per la stima distribuita del campo di interesse, per simulare la comunicazione gossip, ad ogni iterazione, vengono selezionati in maniera casuali due agenti adiacenti dalla matrice Ad, e viene successivamente calcolato l'RMS medio.

---

function DISTRIBUITEDCV(Ad, γ)
  //fase 1
  $C^T Y_{temp} \leftarrow 0$
  $C^T C_{temp} \leftarrow 0$
  for all agents h in Q **do**
    for all $(x_i, y_i) \in Q_h$ **do**
      $p \leftarrow$ random(0,1)
      if $p < 0.8$ **then**
        $C^T Y_{temp} \leftarrow C^T Y_{temp} + s(x_i)^T y_i$
        $C^T C_{temp} \leftarrow C^T C_{temp} + s(x_i)^T s(x_i)$
        x_i viene inserito in TS
      else
        x_i viene inserito in VS
    $C^T Y_h \leftarrow C^T Y_{temp}$
    $C^T C_h \leftarrow C^T C_{temp}$
  // fase 2
  $\hat{y} \leftarrow DistEstimation(Ad, C^T C, C^T y, \gamma)$
  // fase 3
  for all agents h in Q **do**
    $somma \leftarrow 0$
    for all $(x_i, y_i) \in VS_h$ **do**
      $somma \leftarrow somma + (\hat{y}_i - y_i)^2$
    $N \leftarrow length(VS_h)$
    $RMS[h] \leftarrow sqr(somma \backslash N)$
  // fase 4
  $n \leftarrow 0$
  while $n < 100$ **do**
    $h \leftarrow$ agente nella mappa Q
    $k \leftarrow$ agente in Ad[h]
    $temp_h \leftarrow RMS[h]$ //variabile temporanea
    $temp_k \leftarrow RMS[k]$ //variabile temporanea
    $RMS[h] \leftarrow (RMS[h] + RMS[k])/2$
    $RMS[k] \leftarrow RMS[h]$
    if $temp_h = RMS[h]$ **then**
      $n \leftarrow n + 1$
    else
      $n \leftarrow 0$
    if $temp_k = RMS[k]$ **then**
      $n \leftarrow n + 1$
    else
      $n \leftarrow 0$
  return $RMS[1]$ // gli altri sono uguali

2.2.4 Confronto tra CV Centralizzata e CV Distribuita

Come visto nelle sezioni precedenti, adottare metodi distribuiti al posto di quelli centralizzati ha dei grossi vantaggi, e, per alcuni problemi, è l'unica soluzione possibile. In questa sezione ci si pone il problema di verificare se le prestazioni del sistema risentano o meno dell'utilizzo di più agenti in maniera distribuita ed, eventualmente, quan-

tificare questo peggioramento. Per prima cosa si effettua un confronto tra la curva ottenuta tramite la CV centralizzata e quella ottenuta tramite la CV distribuita. É subito chiaro che, nel primo caso, si ottiene una stima mediamente migliore, infatti i valori di RMS sono leggermente più bassi.

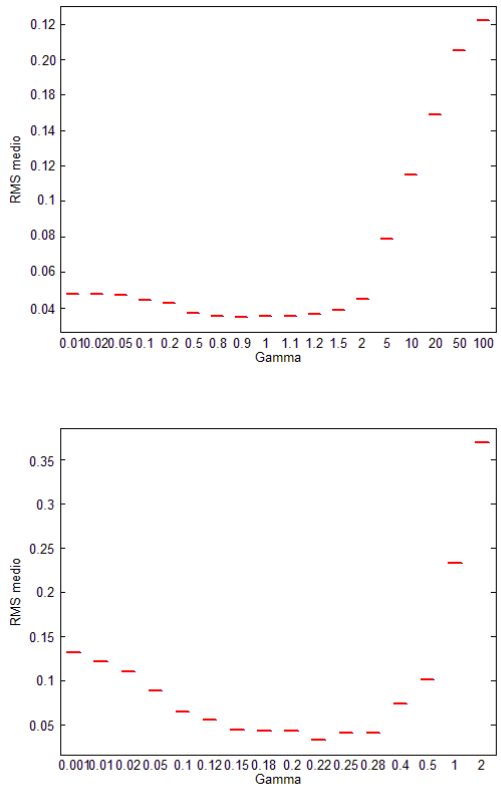

Figura 22: CV Centralizzata e CV Distribuita.

Il confronto appena fatto non può essere esaustivo, infatti si basa su una sola simulazione per ogni γ e non ci può dare, un'idea di come si comporti il sistema mediamente. Per ottenere dei risultati soddisfacenti, si è deciso di compiere le seguenti simulazioni:

1. consideriamo una serie di prove in cui viene fatta aumentare la dimensione della mappa, mantenendo costante il numero di misure complessivo;
2. per ognuna di queste prove se eseguono gli algoritmi di CV un numero di volte sufficientemente grande (nel caso implementato 100), ottenendo quindi una serie di γ ottimi sia nel caso centralizzato ($\gamma_{ott,centr}$) che in quello distribuito($\gamma_{ott,distr}$);
3. per ognuno di questi γ ottimi viene calcolato l'RMS e, per ogni prova, viene calcolato l'RMS medio del caso centralizzato e di quello distribuito.

Remark 15 Come ci si poteva aspettare, l'RMS aumenta mediamente all'aumentare delle dimensioni del problema, poichè i punti in cui vengono effettuate le misure, che corrispondono ai centri delle celle, sono via via più distanti e il rapporto tra numero di misure e numero di punti della mappa diminuisce. Confrontando i due metodi, inoltre, si nota che la CV nel caso centralizzato porta ad avere un errore di stima minore di quello distribuito, anche se i due andamenti sono piuttosto vicini tra loro.

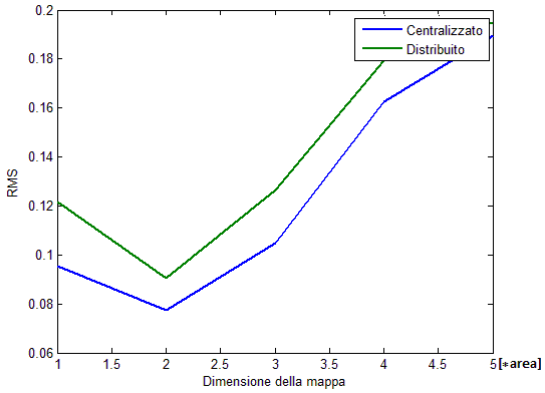

Figura 23: Confronto tra CV Centralizzata e CV Distribuita.

2.3 Effetto del Rumore di Misura

Nella fase di acquisizione del campo di interesse, i sensori utilizzati dai vari agenti introducono un rumore, che è ovviamente la causa degli errori di stima. Il rumore di misura viene generalmente rappresentato da un processo di rumore bianco, caratterizzato, cioè dall'avere una densità spettrale di potenza costante a tutte le frequenze. Per simulare sistemi di misur,a con incertezza diversa è, quindi, sufficiente variare la varianza del rumore bianco. Dalla figura 24 si nota chiaramente che l'errore di stima aumenta linearmente con la varianza del rumore di misura.

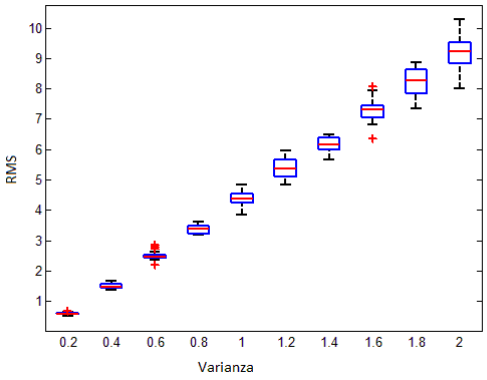

Figura 24: Errore di stima in funzione della varianza del rumore.

3 Stima Distribuita e Aggiornamento del Partizionamento

Una volta che è stato determinato il partizionamento ideale senza conoscenze sul campo da stimare, è possibile modificare la disposizione degli agenti sfruttando le misure che acquisiscono in fase di esplorazione. Questo procedimento ha alcuni importanti vantaggi. Per prima cosa, utilizzando le prime misure ottenute, è possibile intuire immediatamente le regioni in cui il campo è maggiormente rilevante, concentrando in breve tempo gli agenti in quelle aree, ed ottenendo un numero maggiore di misure utili per stimarlo. Altro vantaggio sta nel fatto che, aggiornando continuamente le regioni assegnate ad ogni agente, è sempre possibile migliorare la loro disposizione che, sopratutto nelle fasi iniziali, potrebbe non essere molto efficace. Dal momento che gli algoritmi sviluppati per il partitioning e per la stima distribuita sono entrambi di tipo gossip, per ottenere un partitioning aggiornato ad ogni iterazione dalla stima del campo si è deciso di creare un nuovo algoritmo gossip in cui, ad ogni iterazione, i due agenti che si mettono in comunicazione, calcolano il vettore di informazione locale sommando il termine relativo all'ultima misura acquisita. Successivamente essi effettuano una media del vettore di informazione (25) per ottenere una stima comune del campo, e determinano il partizionamento di Voronoi dell'unione delle due regioni interessate, pesando i singoli punti con il valore assunto in essi dalla stima appena calcolata (6).

---

function DistEstPartitioning(Ad, C^TC, C^Ty, γ)
  $n \leftarrow 0$
  while $n < 100$ **do**
    $h \leftarrow$ agente nella mappa Q
    $k \leftarrow$ agente in Ad[h]
    h e k acquisiscono una misura
    $C^TC_h, C^TC_k \leftarrow (C^TC_h + C^TC_k)/2$
    $C^TY_h, C^TY_h \leftarrow (C^Ty_h + C^Ty_k)/2$
    $\hat{\theta} \leftarrow (C^TC + \gamma I)^{-1}C^TY$
    $\hat{y} \leftarrow s\hat{\theta}$
    $V_i^* \leftarrow V_i(t)$
    $V_j^* \leftarrow V_j(t)$
    $c_i \leftarrow C(V_i(t))$
    $c_j \leftarrow C(V_j(t))$
    $W \leftarrow V_i \cup V_j$
    S:= lista di tutte le coppie di celle di W
    for all (a, b) $\in$ S **do**
      $V_a \leftarrow \{x \in W : d_W(x,a) \leq d_W(x,b)\}$
      $V_b \leftarrow \{x \in W : d_W(x,a) > d_W(x,b)\}$
      if $H_{one}(a, V_a) + H_{one}(b, V_b) <$
        $H_{one}(c_i, V_i^*) + H_{one}(c_j, V_j^*)$ **then**
        $V_i^* \leftarrow V_a$
        $V_j^* \leftarrow V_b$
        $c_i \leftarrow a$
        $c_j \leftarrow b$
      $V_i(t+1) \leftarrow V_i^*$
      $V_j(t+1) \leftarrow V_j^*$
    if $V_i(t+1) = V_i(t)$ **then**
      $n \leftarrow n + 1$
    else
      $n \leftarrow 0$
    if $V_j(t+1) = V_j(t)$ **then**
      $n \leftarrow n + 1$
    else
      $n \leftarrow 0$

Per rendere più visibile il comportamento dell'algoritmo si è scelto di considerare un campo concentrato in una parte della mappa, ottenendo, dopo un numero piuttosto ridotto di iterazioni, delle regioni molto estese dove il campo ha una ampiezza bassa, e regioni più piccole e concentrate nei punti in cui il campo assume valori maggiori (Figura 25).

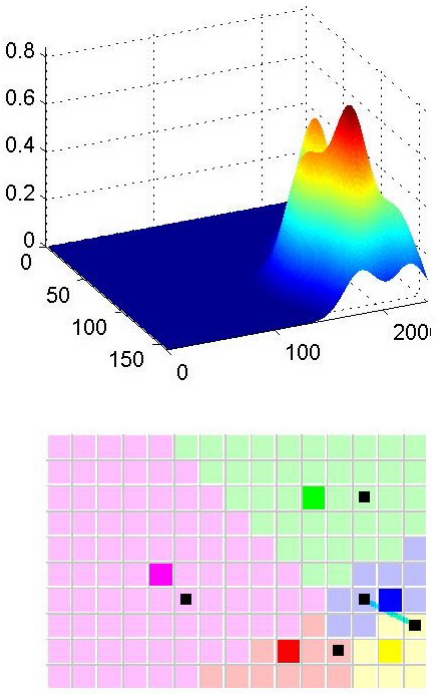

Figura 25: Partizionamento pesando i punti con la stima di un campo.

Quando il partizionamento non varia per un numero sufficientemente elevato di iterazioni è possibile supporre che anche la stima del campo di interesse si pressoché costante, infatti, se così non fosse, il peso associato ai punti della mappa varierebbe e con esso anche il partizionamento. Per verificare ulteriormente la capacità di adattamento del partitioning alla stima, si è successivamente spostato il campo di interesse, ottenendo un rapido spostamento degli agenti verso di esso (Figura 26).

Infine, viene effettuata una simulazione con un campo disposto in più parti della mappa, ottenendo un partizionamento in cui, le regioni centrali dove il campo è pres-

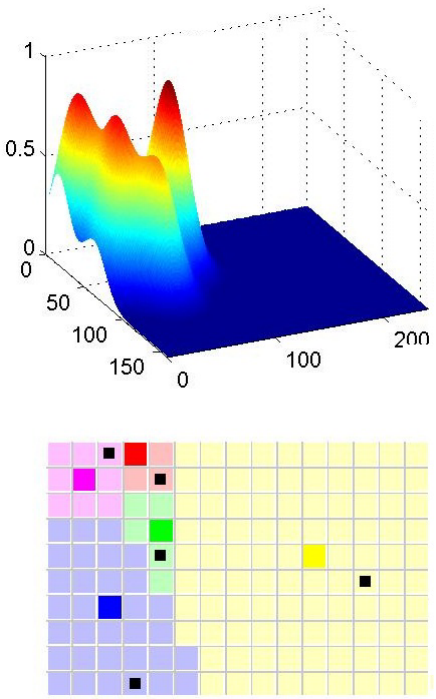

Figura 26: Partizionamento pesando i punti con la stima di un campo.

soché nullo sono estese, mentre le regioni che corrispondono ai picchi del campo sono molto più piccole e gli agenti impiegano meno tempo ad effettuare delle misure dove effettivamente sono necessarie.

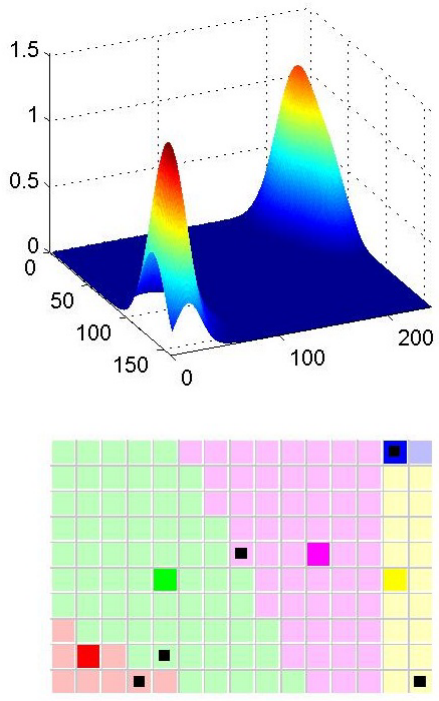

Figura 27: Ripartizionamento con campo distribuito in più regioni.

Conclusioni e Sviluppi Futuri

Nel lavoro svolto si è cercato di risolvere problemi di stima e di partitioning, implementando algoritmi centralizzati già ampliamente trattati in letteratura e sviluppando algoritmi analoghi, ma in un contesto distribuito. Per quanto riguarda il partitioning, non si sono notate grosse differenze nel risultato finale, infatti le regioni ottenute utilizzando una unità centrale con a disposizione le informazioni relative a tutte le celle, corrispondono con buona approssimazione a quelle ottenute tramite l'algoritmo del consensus. Anche per quanto riguarda il tempo di esecuzione dei due algoritmi, si ha che, in entrambi i casi, il numero di iterazioni necessarie a ottenere un partizionamento ottimo, è dello stesso ordine di grandezza del numero di celle considerate. Si è quindi ottenuto un algoritmo in grado di compiere il partizionamento ottimo con prestazioni molto simili a quelle del caso centralizzato, ma con tutti i vantaggi che consegue all'utilizzo di un insieme distribuito di agenti.

Anche per la stima del campo di interesse, si è notato che i due algoritmi portano ad avere risultati piuttosto simili, infatti l'RMS della stima distribuita è di poco superiore a quello della stima centralizzata. Utilizzando il parametro di regolarizzazione in entrambi i casi, inoltre, è possibile migliorare la stima, seppur in maniera modesta, e si eliminano completamente i problemi dovuti alla presenza di quasi singolarità. La stima del parametro ottimo, è stata quindi implementata anche in maniera distribuita, ed ha portato ad ottenere un γ ottimo con errore di stima molto vicino a quello trovato nel caso centralizzato.

Nella determinazione del parametro di regolarizzazione ottimo, si sono riscontrate alcune difficoltà soprattutto quando si è cercato di verificare il comportamento dell'algoritmo al variare della dimensione del problema. Il γ ottimo, infatti, non rimane sempre lo stesso, e sarebbe quindi necessario variare in maniera dinamica il range di valori considerati per ogni prova. Un possibile sviluppo futuro, quindi, potrebbe essere sviluppare un algoritmo in grado di determinare in maniera autonoma il parametro di regolarizzazione, anche variando il numero di misure a disposizione degli agenti, o la dimensione della mappa da esplorare. Un altro argomento su cui lavorare è la scelta della complessità del modello. Nel progetto, infatti, si è supposto di conoscere il numero di parametri da utilizzare nella stima, oppure sono stati provati alcuni valori scelti a caso per verificare gli effetti sulla stima. Utilizzando alcuni criteri come il Final Prediction Error (FPE) [28], l' Akaike Information Criterion (AIC) o il Minimum Description Length (MDL), è possibile stimare la complessità del modello e determinare il miglior numero di parametri da considerare.

Infine, è stato sviluppato un algoritmo che permette di sfruttare le misure ottenute dagli agenti per effettuare

nuovamente il partizionamento, in modo tale da concentrare le regioni nei punti in cui è maggiormente rilevante. Sarebbe interessante anche determinare il gradiente del campo, per poter concentrare gli agenti dove il campo varia più velocemente ed, in genere, è maggiore l'errore di predizione.

Appendix

Definizione di Distanza

Per il calcolo della distanza tra due celle è necessario tenere conto delle presenza di ostacoli. In tal caso, infatti, si verificano delle situazioni in cui non è presente un segmento diretto che congiunge le celle, ed è necessario aggirare gli ostacoli che si interpongono per calcolare la distanza. Si arriva allora alla seguende definizione della distanza.

Definition 16 *(distanza) Dato un ambiente discretizzato Q e due celle $a, b \in Q$, la distanza tra queste due celle è data da:*

1. *in assenza di ostacoli: $d(a,b) = \sqrt{x_b^2 - x_a^2}$, dove x_a e x_b sono rispettivamente le coordinate dei centri delle celle a e b;*
2. *in presenza di ostacoli: sia $x_1, ..., x_k$ la più breve sequenza di celle che congiunge x_a e x_b, allora la distanza tra x_a e x_b è data da:*

$$d(x_1, x_k) = \sum_{i=1}^{k-1} d(x_i x_{i+1}) \tag{36}$$

è possibile restringere il calcolo della distanza alle celle di una regione V di Q.

Definition 17 *(distanza vincolata) Dato un ambiente discretizzato Q, una regione $V \subset Q$ e due celle $a, b \in V$, la distanza tra queste due celle è data da:*

1. *in assenza di ostacoli: $d_V(a,b) = \sqrt{x_b^2 - x_a^2}$, dove x_a e x_b sono rispettivamente le coordinate dei centri delle celle a e b;*
2. *in presenza di ostacoli: sia $x_1, ..., x_k \in V$ la più breve sequenza di celle di V che congiunge x_a e x_b, allora la distanza tra x_a e x_b è data da:*

$$d_v(x_1, x_k) = \sum_{i=1}^{k-1} d_v(x_i x_{i+1}) \tag{37}$$

Dimostrazione Formula MQ

Si definisce una cifra di merito

$$V(\theta) := \sum_{i=1}^{N_Q} [y_i - \hat{y}_i(\theta)]^2 \tag{38}$$

il modello che meglio descrive i dati osservati è quello che corrisponde al valore $\hat{\theta}$, per cui $V(\hat{\theta})$ è minimo. Ovvero

$$V(\hat{\theta}) = \min V(\theta) \tag{39}$$

la funzione $\hat{\theta}$ si chiama *stimatore ai minimi quadrati di θ*.

La minimizzazione di $V(\theta)$ diviene allora il problema di minimizzare la norma:

$$V(\theta) = \|\mathbf{y} - \mathbf{C}\theta\|^2.$$

Si tratta di determinare la proiezione ortogonale di y sullo spazio $\boldsymbol{S} = span(\mathbf{C})$, quindi $\hat{\theta}$ è l'unico vettore θ tale che

$$\mathbf{S} \perp (\mathbf{y} - \mathbf{C}\theta) \tag{40}$$

in altre parole

$$\mathbf{C}^T\mathbf{y} - \mathbf{C}^T\mathbf{C}\theta = 0 \Longrightarrow \hat{\theta}(\mathbf{y}) = [\mathbf{C}^T\mathbf{C}]^{-1}\mathbf{C}^T\mathbf{y} \tag{41}$$

$$\hat{\theta}(\mathbf{y}) = \left[\sum_{i=1}^{N_Q} s(x_i)^T s(x_i)\right]^{-1} \sum_{i=1}^{N_Q} s(x_i) y_i \tag{42}$$

da cui si vede che lo stimatore è sempre funzione lineare delle misure.

Riferimenti bibliografici

[1] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer, Special Issue in Sensor Networks*, vol. 37, pp. 41–49, 2004. 1

[2] S. Oh, L. Schenato, P. Chen, and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: System design, algorithms and experiments," in *Proceedings of the IEEE*, 2007. 1

[3] A. Singh, M. Batalin, M. Stealey, V. Chen, Y. Lam, M. Hansen, T. Harmon, G. Sukhatme, and W. Kaiser, "Mobile robot sensing for environmental applications," in *In Proceedings of the International Conference on Field and Service Robotics*, 2007. 1

[4] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stability of flocking motion," University of Pennsylvania, Tech. Rep., 2003. 1

[5] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, pp. 655–661, 2005. 1

[6] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithm and theory," *IEEE Transactions on Automatic Control*, vol. 15, pp. 401–420, 2006. 1

[7] A. Jadbabie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, pp. 988–1001, 2003. 1

[8] J. Cortés, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, 2004. 1

[9] J. Adler, "Chemotaxis in bacteria," *Journal of Supramolecular Structure*, vol. 4, pp. 305–317, 1966. 1

[10] A. Dhariwal, G. S. Sukhatmeand, and A. A. G. Requicha, "Bacterium-inspired robots for environmental monitoring," in *Proceedings of the IEEE international conference on robotics and automation*, 2004. 1

[11] "Department of defense - office of naval research - muri - adaptive sampling and prediction project," 2009. 1

[12] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobilesensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Transaction on Automatic Control*, vol. 49, p. 1292, 2004. 1

[13] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: From natural to artificial systems.* Oxford University Press, 1999, vol. –. 1

[14] R. C. Eberhart, Y. Shi, and J. Kennedy, *The Morgan Kaufmann series in artificial intelligence, Swarm intelligence.* Morgan Kaufmann, 2001. 1

[15] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, p. 357, 2009. 1

[16] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *CoRR*, vol. abs/1011.1939, p. 13, 2010. 1, 2, 3, 4, 6, 7

[17] S. Geisser, *Predictive Inference.* New York: Chapman and Hall, 1993. 2

[18] R. Kohavi, "A study of crossvalidation and bootstrap for accuracy estimation and model selection," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann*, 1995. 2

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2008. 2, 12

[20] S. Oh and J. Choi, "Distributed learning in mobile sensor networks using cross validation," *IEEE Conference on Decision and Control December*, vol. 49, pp. 15–17, 2010. 2

[21] R. Carli, "Territory partitioning problems and coverage control algorithms," university of Padova. 3

[22] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the lloyd algorithm for computing centroidal voronoi tessellations," *SIAM Journal on Numerical Analysis*, vol. 44, pp. 102–119, 2006. 3

[23] S. Bolognani, S. D. Favero, L. Schenato, and D. Varagnolo, "Consensus-based distributed sensor calibration and least-square parameter identification in wsns," *International Journal of Robust and Nonlinear Control*, vol. 20, pp. 176–193, 2010. 5

[24] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," *Networked Control Systems*, vol. 406, pp. 75–107, 2010. 5

[25] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 26, pp. 634–649, 2008. 5

[26] F. Bullo, R. Carli, and P. Frasca, "Gossip coverage control for robotic networks: Dynamical systems on the the space of partitions," 2010. 7

[27] X. Xiao, R. Mukkamald, and R. J. Cohen, "A weighted principal component regression approach for system identification," *Statistical Signal Processing*, pp. 206–209, 2003. 12

[28] G. Picci, "Metodi statistici per l'identificazione di sistemi lineari," dispense del corso di Identificazione dei modelli dinamici. 19