

UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze e Tecnologie
Corso di Laurea in Informatica

TEMP IDENTIFICAZIONE DI SATELLITI

Relatore: Prof. Giovanni RIGHINI

Correlatore:

Tesi di:
Jonathan Junior AGYEKUM
Matricola: 935132

Anno Accademico 2022-2023

dedicato a ...

Prefazione

prefazione

Organizzazione della tesi

La tesi è organizzata come segue:

- nel Capitolo 1

Ringraziamenti

Grazie.

Indice

	ii
Prefazione	iii
Ringraziamenti	iv
1 Introduzione	1
1.1 Strumenti utilizzati	1
1.2 Ricerca operativa e astronomia	1
1.3 Storia dell'identificazione di satelliti	1
1.3.1 Rilevamenti diretti e indiretti	1
2 Descrizione del problema	2
2.1 Descrizione	2
2.2 Campi di applicazione e utilità	3
3 Sviluppo del modello	4
3.1 Definizione del modello	4
3.1.1 Calcolo della regione pareto-ottima	7
3.2 Algoritmo di controllo	12
3.2.1 Definizione del passo per il minimo periodo β	12
3.2.2 Criterio arresto	13
3.2.3 Punto di inizializzazione	14
3.2.4 Scelta del solutore	24
3.2.5 Gestione degli errori	26
3.2.6 Selezione della soluzione	27
4 Sviluppo dell'algoritmo	57
4.1 Struttura e approccio	57
4.1.1 Creazione dei sottoproblemi	58
4.1.2 Gestione dei nodi	58

4.1.3	Gestione delle foglie	59
4.1.4	Valutazione delle prestazioni	62
4.1.5	Miglioramenti	68

Capitolo 1

Introduzione

1.1 Strumenti utilizzati

1.2 Ricerca operativa e astronomia

1.3 Storia dell'identificazione di satelliti

1.3.1 Rilevamenti diretti e indiretti

Capitolo 2

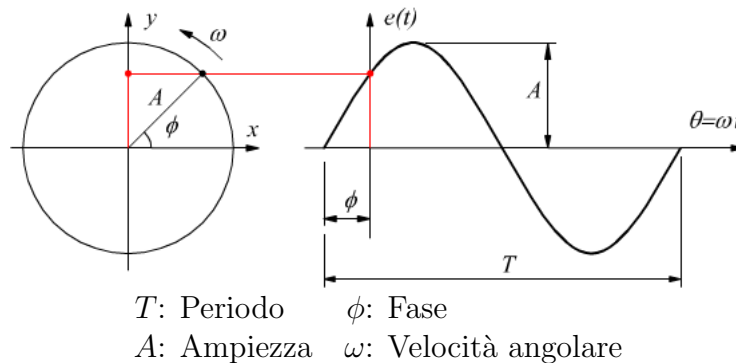
Descrizione del problema

2.1 Descrizione

L'osservazione in modo continuo dell'angolo descritto da k satelliti intorno al loro pianeta, genera delle sinusoidi, questo apre alla possibilità di una identificazione dei satelliti attraverso l'analisi del loro moto orbitale.

Il legame tra l'orbite dei satelliti e le sinusoidi può essere spiegata considerando le orbite dei satelliti circolari e quindi attraverso le leggi del moto circolare.

Figura 1: Moto circolare uniforme



Dove il periodo corrisponde al tempo di rivoluzione del satellite, l'ampiezza alla massima distanza che intercorre nel periodo tra il satellite e il corrispettivo pianeta, $e(t)$ lo spostamento e la distanza del satellite lungo l'orbita, dal pianeta di riferimento. È quindi possibile descrivere la sinusoide risultante tramite l'espressione analitica:

$$e(t) = A \sin(\omega t + \phi)$$

L'obiettivo è quello di riuscire ad attribuire ai k satelliti i k valori di ogni osservazione e di trovare periodo, ampiezza e fase delle k sinusoidi, supponendo di poter osservare i k valori simultaneamente, in n momenti successivi. Si suppone di non conoscere nulla dei k satelliti e soprattutto di non poterli distinguere in fase di rilevazione dei valori.

Il problema è scomponibile in due sottoproblemi: uno di interpolazione e uno di assegnamento. La componente di interpolazione è identificata nell'operazione di individuazione della sinusoide che si scosta il meno possibile dai valori. Questo definisce un problema di interpolazione non-lineare e continuo: non-lineare poiché una sinusoide è una funzione periodica non-lineare, definita da una trasformazione elementare della funzione seno, continuo poiché il dominio di una sinusoide e delle sue componenti è definito in \mathbb{R} o in un suo sottoinsieme.

La componente di assegnamento è individuata nel processo di selezione dei punti tra le diverse osservazioni da interpolare. La funzione di questa componente è quella di individuare k sinusoidi, generate dall'interpolazione di una assegnazione di valori, che rappresentino i k satelliti.

2.2 Campi di applicazione e utilità

Il principio della tipologia di identificazione che ho studiato è quello di riconoscere satelliti di un pianeta in base alla variazione in un determinato dominio di una caratteristica quantificabile, in questo caso l'angolo, in funzione del tempo. La variazione dev'essere riconoscibile e identificativa del satellite, cioè deve definire un andamento caratteristico di una funzione elementare o una combinazione di essi. Questo principio è illustrato, per esempio dal metodo di trasito, metodo di identificazione fotometrico di satelliti extrasolari [4].

La tipologia di identificazione che illustrerò, può essere applicata sia ai casi di rilevamento diretto, tecniche che permettono di osservare direttamente al telescopio i satelliti, e sia indiretti, cioè l'individuazione tramite effetti fisici che satellite può indurre. Il vincolo fondamentale è che sia possibile effettuare misurazioni dell'angolo rispetto al suo pianeta.

L'identificazione attraverso l'analisi del moto orbitale può essere utile in tutte le situazioni dove non è possibile riconoscere o distinguere due o più satelliti o qualunque entità il quale movimento è osservabile e rotante.

Capitolo 3

Sviluppo del modello

3.1 Definizione del modello

Propongo un modello per la risoluzione del sottoproblema di interpolazione. La sinusoidale che si vuole ottenere dall'interpolazione dei punti deve avere determinate caratteristiche. Tra tutte le sinusoidi possibili che interpolano gli n valori, che in questo contesto chiamerò punti, si vuole quella di massimo periodo e sicché le misurazioni possono essere affette da errore si deve combinare la ricerca del massimo periodo con quella di minimo errore, definendo un problema a due obiettivi.

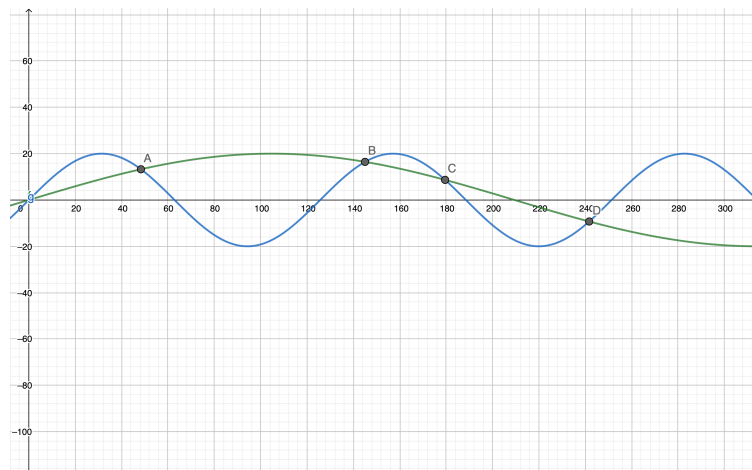
Il processo risolutivo per un problema a molti obiettivi in conflitto tra loro, cioè che il miglioramento di una comporti un peggioramento di un'altra, prevede:

1. Il calcolo della regione pareto-ottima, cioè l'insieme di soluzioni ammissibili non dominate, denominate anche soluzioni paretiane.
2. La scelta di una soluzione tra quelle individuate.

La determinazione della regione pareto-ottima può essere effettuata in diversi modi. I modelli che ho definito inseguito utilizzano rispettivamente metodo dei pesi e metodo dei vincoli, i due modelli si differenziano solo per vincoli e funzione obiettivo.

In questo processo di interpolazione è necessario fare attenzione alle sinusoidi con basso periodo rispetto alla sinusoidale da individuare: si può osservare dalle figure 2, 3 e 4 che aumentando la frequenza, quindi al diminuire del periodo, esiste sempre un modo di interpolare n punti con una sinusoidale minimizzando a piacimento l'errore di interpolazione, poiché i punti possono corrispondere o essere vicini alle intersezioni tra le sinusoidi.

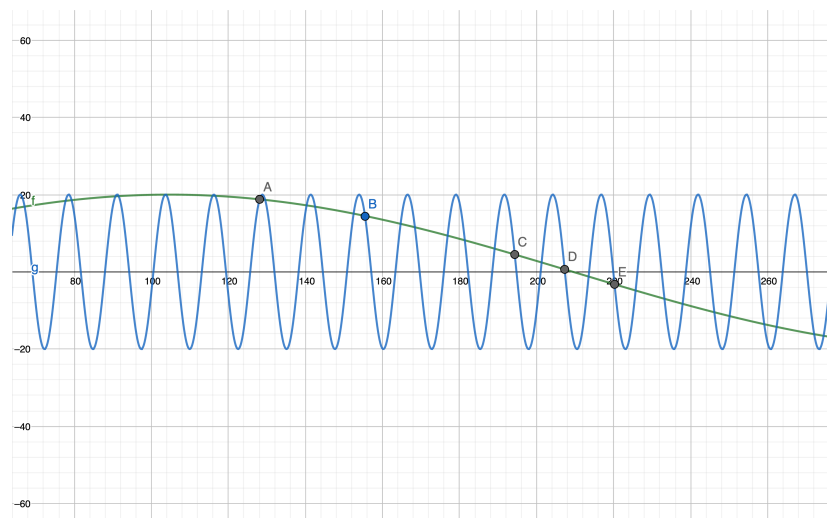
Figura 2: Problema sinuoidi da basso periodo rispetto alla senoide da individuare



Ascisse: Periodo (s)

Ordinate: Distanza del satellite dal pianeta (m)

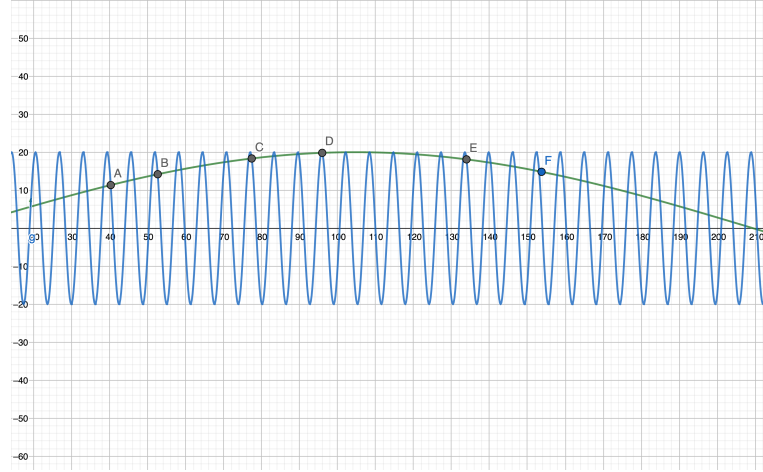
Figura 3: Problema sinuoidi da basso periodo rispetto alla senoide da individuare



Ascisse: Periodo (s)

Ordinate: Distanza del satellite dal pianeta (m)

Figura 4: Problema sinuoidi da basso periodo rispetto alla senoide da individuare



Ascisse: Periodo (s)

Ordinate: Distanza del satellite dal pianeta (m)

Parametri I dati a disposizione del modello sono gli n punti assegnati da interpolare, il quale numero è determinato dal modulo di assegnamento. I punti sono individuati in un piano cartesiano dove l'asse delle ascisse rappresenta il tempo e l'asse delle ordinate l'angolo. I parametri del modello sono quindi le n ascisse t_i e le n ordinate e_i degli n punti, con i da 1 a n .

Per una maggiore semplicità numerica considero i secondi per il tempo mentre rappresento gli angoli in radianti.

Variabili Le variabili sono individuate nelle componenti che determinano una senoide, data l'espressione analitica generale di una funzione sinusoidale:

$$e_i = A \sin(\omega t_i + \phi) \quad (1)$$

Dall'espressione ho definito le variabili che rappresentano fase ϕ , ampiezza A e la pulsazione ω .

Inoltre c'è la necessità di rappresentare l'errore che può essere generato dalle misurazioni o da un assegnamento di punti errato. L'espressione analitica diventa:

$$e_i = A \sin(\omega t_i + \phi) + \varepsilon_i \quad (2)$$

Perciò ho definito n variabili aggiuntive che rappresentano l'errore per ogni punto.

3.1.1 Calcolo della regione pareto-ottima

Metodo dei pesi

Il metodo dei pesi consiste nel dare un peso a ciascuna delle funzioni obiettivo, così da ridurre il problema a molti obiettivi in uno di programmazione matematica parametrica.

Vincoli Ho definito come unico vincolo l'equazione (2)

Funzione obiettivo Le due funzioni obiettivo da ottimizzare sono:

$$\min f_1 = \omega \quad (3)$$

La massimizzazione del periodo

$$\min f_2 = \frac{\sum_{i=1}^n \varepsilon_i^2}{n} \quad (4)$$

La minimizzazione dell'errore complessivo, calcolo l'errore quadratico complessivo per l'eliminazione dei possibili valori negativi che si possono generare.

I pesi definiti per le funzioni f_1 e f_2 sono rispettivamente 1 e -1, quindi ottenendo la funzione obiettivo:

$$\min f = f_1 + f_2 \quad (5)$$

Questa configurazione dei pesi di partenza permette di penalizzare tutte quelle soluzioni ammissibili che massimizzano il periodo ma hanno un errore elevato.

Il processo di individuazione delle soluzioni paretiane, con questo metodo e numero di funzioni, consiste nel eseguire un'analisi parametrica sui pesi, cioè analizzare e individuare soluzioni non dominate al variare dei pesi entro un certo range. In questo caso può essere sufficiente solo effettuare la variazione del peso dell'errore.

Un modello che dà priorità al periodo ottenendo un alto errore è per certo non utilizzabile, perché l'alto errore denota una completa mancanza di relazione tra i punti e la sinusoide individuata, questo spiega l'analisi parametrica solo sul peso dell'errore. È anche possibile non effettuare alcuna variazione dei pesi, ed accontentarsi della prima soluzione individuata tramite la configurazione dei pesi di partenza.

Metodo dei vincoli

Il metodo dei vincoli consiste nell'ottimizzare una delle funzioni obiettivo, trasformando le rimanenti in vincoli, utilizzando un termine noto parametrico.

Vincoli Si ha l'equazione (2) e la trasformazione in vincolo della funzione obiettivo legato al periodo (3)

$$\frac{2\pi}{\omega} \geq \beta \quad (6)$$

Il termine noto β sta ad indicare il minimo periodo ammissibile per una soluzione, questo permette di escludere le soluzioni a basso periodo rendendo inammissibili soluzioni con periodo più piccolo del termine noto.

Oppure se si applica il metodo dei vincoli sulla funzione obiettivo sull'errore (4) si ottiene:

$$\frac{\sum_{i=1}^n \varepsilon_i^2}{n} \leq \alpha \quad (7)$$

Il termine noto α rappresenta il massimo errore ammissibile per una soluzione.

Funzione obiettivo La funzione obiettivo definita è la funzione (4).

L'individuazione delle soluzioni paretiane è effettuata variando il termine noto parametrico nel vincolo (6). Ho scelto di definire il vincolo (6) perché permette di avere un controllo migliore sull'esplorazione dei periodi possibili della sinusoide da individuare, data dalla possibilità di definire come soluzione di partenza del modello, un assegnazione della variabile che rappresenta la pulsazione a partire dal termine noto β .

$$\omega = \frac{2\pi}{\beta} \quad (8)$$

Questa scelta risulta preferibile anche a seguito di sperimentazioni. Eseguendo i solutori con un modello definito con il metodo dei vincoli sull'errore e un modello con il metodo dei vincoli definito sul periodo, dai seguenti problemi e punti di inizializzazione si ottiene:

- Problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con ω pari a $0.0013 \frac{rad}{s}$ che corrisponde ad un periodo di 4830s, con t_i che assume i valori nell'insieme intero $[0, 9]$

Tabella 1: Prestazioni dei solutori con modello con vincolo sull'errore

Solutore	soluzione $p_{init} = 2500s$		soluzione $p_{init} = 3694s$		soluzione $p_{init} = 4186s$	
	periodo (s)	errore (n)	periodo(s)	errore (n)	periodo (s)	errore (n)
COBYLA	6e6	1.0e-3	6e6	3.0e-4	6e6	3.7e-4
BOBYQA	14e9	4.6e-5	14e9	4.6e-5	14e9	4.6e-5
NEWUOA	6e6	4.8e-5	6e6	4.8e-5	6e6	4.8e-5
PRAXIS	2e5	4.8e-5	2e4	4.1e-5	5e5	5.1e-5
SBPLX	3e5	3.7e-5	3e5	3.7e-5	3e5	3.7e-5

p_{init} rappresenta il periodo di inizializzazione da cui si ricava la pulsazione da fissare alla variabile ω , per ottenere una soluzione iniziale.

Tabella 2: Prestazioni dei solutori con modello con vincolo sul periodo

Solutore	soluzione $p_{init} = 2500s$		soluzione $p_{init} = 3694s$		soluzione $p_{init} = 4186s$	
	periodo (s)	errore (n)	periodo(s)	errore (n)	periodo (s)	errore (n)
COBYLA	4244	2.4e-5	4477	1.4e-5	4185	1.6e-5
BOBYQA	7324e5	3.4e-5	9645e3	3.1e-5	2708e6	3.4e-5
NEWUOA	3980e-7	0.0	2707e-7	0.0	2388e-7	0.0
PRAXIS	3982e-6	3e-1	2710e-6	6.3	2387e-6	6.2
SBPLX	inf		inf		4188	1.3e-15

p_{init} rappresenta il periodo di inizializzazione da cui si ricava la pulsazione da fissare alla variabile ω , per ottenere una soluzione iniziale.

- Problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con ω pari a $0.8 \frac{rad}{s}$ che corrisponde ad un periodo di 7.8s con t_i intero da 0 a 9 che assume i valori nell'insieme intero $[0, 9]$

Tabella 3: Prestazioni dei solutori con modello con vincolo sull'errore

Solutore	soluzione $p_{init} = 2500s$		soluzione $p_{init} = 1250s$		soluzione $p_{init} = 500s$	
	periodo (s)	errore (n)	periodo(s)	errore (n)	periodo (s)	errore (n)
COBYLA	514.7	0.2	230.0	0.2	184.5	0.2
BOBYQA	787.6	0.4	870.6	0.3	870.6	0.3
NEWUOA	999.9	0.0	999.9	0.0	999.9	0.0
PRAXIS	999.9	0.0	999.9	0.0	999.9	0.0
SBPLX	inf		inf		inf	

p_{init} rappresenta il periodo di inizializzazione da cui si ricava la pulsazione da fissare alla variabile ω , per ottenere una soluzione iniziale.

Tabella 4: Prestazioni dei solutori con modello con vincolo sul periodo

Solutore	soluzione $p_{init} = -2500s$		soluzione $p_{init} = -1250s$		soluzione $p_{init} = -500s$	
	periodo (s)	errore (n)	periodo(s)	errore (n)	periodo (s)	errore (n)
COBYLA	0.8	0.2	1.1	0.3	0.1	0.1
BOBYQA	0.003	0.09	0.007	0.1	0.02	0.1
NEWUOA	0.0004	$4.8e-23$	0.0008	$3e-21$	0.002	$7.4e-19$
PRAXIS	0.0004	0.2	0.0008	0.02	0.002	15 894
SBPLX	0.0008	$1.7e-12$	0.001	$1e-12$	0.004	$3.7e-11$

p_{init} rappresenta il periodo di inizializzazione da cui si ricava la pulsazione da fissare alla variabile ω , per ottenere una soluzione iniziale, inoltre si fissano periodi negativi a causa del vincolo 6 che vincola la ricerca a periodi maggiori del termine noto β

Dai risultati delle sperimentazioni si può notare che nel primo problema, il solutore COBYLA utilizzato con il vincolo sul periodo individua soluzioni che si scostano mediamente di 529s, a differenza se utilizzato con il vincolo sull'errore con gli stessi punti di inizializzazione si ottengono soluzioni dal periodo molto più distanti. Le stesse osservazioni possono essere effettuate nel secondo problema. Invece il restante dei solutori non ottiene soluzione accettabile con nessuno dei due modelli.

Per le sperimentazioni effettuate ho fissato il termine noto α nel vincolo dell'errore (7) a $1e-3$, anche provando a diminuire il valore del termine noto, quindi vincolare il solutore ad individuare soluzioni dall'errore quadratico medio entro il valore fissato, i risultati risultano peggiori rispetto all'utilizzo del modello con metodo dei vincoli applicato alla funzione obiettivo del periodo, da come si può osservare dalle tabelle 5 e 6.

A seguito della diminuzione del limite d'errore, la gran parte dei solutori non riesce ad individuare una soluzione ammissibile, questo può essere causato dal fatto che le soluzioni di inizializzazione siano inammissibili, aggiungendo il necessario compito di individuare per ogni problema soluzioni di inizializzazione ammissibili, per permettere una eventuale ricerca più fruttuosa.

Nonostante questi svantaggi si può notare che con il solutore COBYLA nel problema di tabella 3, il periodo della soluzione individuata diminuisce al diminuire del periodo della soluzione di inizializzazione avvicinandosi in maniera costante al periodo della sinusoide del problema. Questo suggerisce una direzione di ottimizzazione corretta, che sarebbe possibile individuare tramite un algoritmo iterativo se valesse per tutti o per la maggior parte dei casi.

- Problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con ω pari a 0.0013 rad/s che corrisponde ad un periodo di 4830s, con t_i che assume i valori nell'insieme intero $[0, 9]$

Tabella 5: Prestazioni dei solutori con modello con vincolo sull'errore e $\alpha = 1e-6$

Solutore	soluzione $p_{init} = 2500s$		soluzione $p_{init} = 3694s$		soluzione $p_{init} = 4186s$	
	periodo (s)	errore (n)	periodo(s)	errore (n)	periodo (s)	errore (n)
COBYLA	6e6	$1.4e-5$	6e6	$2.7e-4$	6e6	$2.9e-4$
BOBYQA	24e3	$1.5e-5$	21e3	$1.5e-5$	24e3	$1.5e-5$
NEWUOA	1e6	0.0	1e6	0.0	1e6	0.0
PRAXIS	1e6	0.0	1e6	0.0	1e6	0.0
SBPLX	8e5	$4.0e-5$	8e5	$4.0e-5$	8e5	$4.0e-5$

p_{init} rappresenta il periodo di inizializzazione da cui si ricava la pulsazione da fissare alla variabile ω , per ottenere una soluzione iniziale.

- Problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con ω pari a 0.8 rad/s che corrisponde ad un periodo di 7.8s con t_i intero da 0 a 9 che assume i valori nell'insieme intero $[0, 9]$

Tabella 6: Prestazioni dei solutori con modello con vincolo sull'errore e $\alpha = 1e-6$

Solutore	soluzione $p_{init} = 2500s$		soluzione $p_{init} = 1250s$		soluzione $p_{init} = 500s$	
	periodo (s)	errore (n)	periodo(s)	errore (n)	periodo (s)	errore (n)
COBYLA	356.2	0.2	266.0	0.2	101.5	0.2
BOBYQA	173.6	0.4	173.6	0.4	173.6	0.4
NEWUOA	1e6	0.0	1e6	0.0	1e6	0.0
PRAXIS	1e6	0.0	1e6	0.0	1e6	0.0
SBPLX	339	0.3	354	0.3	339	0.3

p_{init} rappresenta il periodo di inizializzazione da cui si ricava la pulsazione da fissare alla variabile ω , per ottenere una soluzione iniziale.

Dati i risultati ed il miglior controllo sull'inizializzazione e sul vincolo 6, ho scelto di utilizzare il modello con il metodo dei vincoli applicato al periodo.

3.2 Algoritmo di controllo

Il modello che ho scelto di implementare è il modello che utilizza il metodo dei vincoli per i vantaggi citati nella sua specificazione. Una volta definito l'implementazione del modello è necessario specificare un algoritmo di controllo e di esecuzione di esso. I compiti sono la definizione del valore del minimo periodo β , nel vincolo (6) ad ogni iterazione e la gestione di eventuali errori.

3.2.1 Definizione del passo per il minimo periodo β

Il passo definisce la quantità aggiunta al minimo periodo ad ogni iterazione. Per facilitare l'individuazione di soluzioni differenti, è possibile definire un passo variabile, così da spostare il punto di inizializzazione del problema dalla regione di un minimo locale, in base alla soluzione ottenuta.

L'algoritmo va ad aumentare il passo ogniquale volta il periodo della soluzione individuata in una iterazione, appartiene ad un intorno del periodo della soluzione dell'iterazione precedente (riga 8), altrimenti si ha un ripristino del passo ad un valore di default (riga 11). In base alle necessità è possibile definire velocità differenti di variazione del passo, tramite la definizione della funzione f (riga 9). Per aumentare il numero di soluzioni individuate è necessario diminuire la velocità di variazione di passo in modo da ottenere inizialmente piccoli spostamenti del minimo periodo, al contrario per ottenere meno soluzioni ed una esplorazione più veloce, è necessario

Algoritmo 1: Algoritmo di controllo del modello

input : *pt* Collezione di punti da interpolare $[(t_1, e_1), \dots, (t_k, e_k)]$
output: *soluzioni* Insieme di soluzioni che interpolano i punti *pt*

```

1 passo  $\leftarrow 1$ 
2  $\beta \leftarrow \text{periodoIniziale}$ 
3 soluzioni  $\leftarrow \emptyset$ 
4 while  $\beta \leq \text{periodoLimite}$  do
5   solPrecedente  $\leftarrow \text{sol}$ 
6   sol  $\leftarrow \text{Interpola}(\text{pt}, \beta, \text{solutore})$ 
7   Aggiungi sol a soluzioni
8   if sol[periodo] = solPrecedente[periodo] then
9     | passo = f(passo)
10  else
11    | passo  $\leftarrow 1$ 
12  end
13   $\beta \leftarrow \beta + \text{passo}$ 
14 end
```

aumentare la velocità di variazione del passo. esempi possono essere:

$$f(x) = x * k \quad (9)$$

Si moltiplica il passo *p* per una costante *k*, è possibile variare la velocità della variazione del passo aumentando o diminuendo il parametro *k*. Si misura la velocità della funzione di variazione del passo tramite la derivata della funzione $f(x)$.

Un altro fattore che determina l'esplorazione è la dimensione dell'intorno. È necessario definire un intorno poiché è possibile che con approssimazioni e specifiche interne del solutore o anche in base alle caratteristiche della regione ammissibile, non si individui l'esatto valore di un minimo locale, ma un valore nell'intorno di esso in iterazioni successive. Perciò è necessario definire un livello di tolleranza con il quale definire se due soluzioni sono da considerare uguali (riga 8).

3.2.2 Criterio arresto

È necessario specificare un criterio di arresto per l'algoritmo di controllo. L'idea applicata è quella di determinare l'arresto fissando un periodo limite per il minimo periodo β (riga 4). Effettuando una ricerca sui satelliti del sistema solare, ho individuato che il massimo periodo di un satellite nel sistema solare è pari a 9374.0 giorni [3].

Quindi è possibile fissare un periodo limite, effettuando un mappaggio da giorni in secondi, pari a 5000s. Non è necessario fissare esattamente il valore estratto dallo studio dei periodi del sistema solare, poiché il periodo β , utilizzato nel vincolo (6) come limite inferiore, non impedisce al solutore di individuare periodi dai valori molto più grandi rispetto ai periodi del sistema solare, tutto questo facendo partire l'individuazione delle soluzioni da un periodo pari a 0, escludendo i periodi negativi. L'esclusione dei periodi negativi non influisce sull'individuazione delle soluzioni poiché i punti sono rilevati all'avanzare del tempo.

3.2.3 Punto di inizializzazione

Il processo di inizializzazione di un modello, consiste nel fissare le variabili del problema a dei valori di partenza. Generalmente definire una soluzione iniziale per il solutore nei problemi di programmazione non lineare è importante, poiché ne determina l'abilità di individuare un minimo locale diverso da un altro, e poter individuare un minimo locale migliore sotto una determinata caratteristica.

In questo problema la qualità della soluzione è individuata prevalentemente nel periodo e nell'errore, perciò la variabile che assume più importanza nella determinazione di una soluzione iniziale è la pulsazione ω , gli errori per ogni punto sono variabili risultanti perciò non vanno fissate.

Con questa definizione del modello, mi è risultato difficile discernere un algoritmo di definizione della soluzione di partenza, a partire dalle sperimentazioni effettuate. La difficoltà nasce nell'interpretare il comportamento del solutore dato un problema, quindi dall'individuazione di un pattern chiaro che definisca la posizione ideale della soluzione di partenza, nelle successive iterazioni.

Le sperimentazioni sono state effettuate eseguendo il modello dando come soluzione iniziale il seguente:

- $A = 0$
- $\phi = 0$
- $w = \frac{2*\pi}{periododipartenza}$

Da come si può notare dalle tabelle 8, 7 e 9, la tendenza dei periodi delle soluzioni rispetto ai periodi di partenza è crescente se il periodo di partenza è inferiore al periodo del problema, invece decrescente se maggiore, con qualche eccezione.

Gli errori rimangono tendenzialmente costanti, nello stesso ordine di grandezza fino all'avvicinarsi del periodo di partenza rispetto al periodo della soluzione. Queste osservazioni possono indurre alla definizione iterativa in maniera crescente o decrescente del periodo di partenza, in base al periodo della soluzione precedente, questa modalità però presenta determinate situazioni particolari che impediscono di avvicinarsi al minimo locale con periodo più vicino al periodo della soluzione:

Tabella 7: Problema di 30 punti generati dalla sinusoide $e_i = 1 \sin(\omega t_i)$ con $\omega = 0.0125 \text{ rad/s}$, perciò periodo di 502.4s

Periodo di partenza (s)	Periodo soluzione (s)	Errore quadratico medio (m^2)
3000	1887.2	2.3e-3
2500	1540.5	1.2e-4
2000	1598.2	3.2e-4
1500	1444.6	6.9e-4
1000	499.9	0.0
100	99.9	1.8e-3
50	260.4	6.0e-3
10	8.6	1.2e-2
1	1.0	1.1e-2

Tabella 8: Problema di 30 punti generati dalla sinusoide $e_i = 1 \sin(\omega t_i)$ con $\omega = 0.0013 \text{ rad/s}$, perciò periodo di 4830.7s

Periodo di partenza (s)	Periodo soluzione (s)	Errore quadratico medio (m^2)
6000	5896.9	3.8e-6
4000	3999.9	0.0
3500	3837.5	1.2e-4
3000	2869.3	1.1e-4
2500	3036.6	1.2e-4
2000	1926.1	1.9e-5
1250	1281.8	2.4e-5
750	750.0	1.4e-4
100	131.6	7.1e-5

Tabella 9: Problema di 30 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.8 \text{ rad/s}$, perciò periodo di 7.85s

Periodo di partenza (s)	Periodo soluzione (s)	Errore quadratico medio (m^2)
2500	2429.7	4.8e-1
2000	2261.6	4.8e-1
1500	1344.5	4.8e-1
1000	1337.6	4.8e-1
500	497.8	4.9e-1
100	113.07	4.9e-1
50	50.2	4.9e-1
10	7.85	7.6e-4
1	7.85	7.1e-5

1. Minimi locali tra il periodo di partenza ed il periodo della soluzione
2. L'errore rimane tendenzialmente costante nonostante le distanze tra il periodo della soluzione individuata e il periodo del problema aumenti.

Definendo un algoritmo (2) che iterativamente che varia il periodo di partenza in base al periodo della soluzione individuata nell'iterazione precedente, si definisce un algoritmo con lo scopo di avvicinare il periodo di inizializzazione ad un minimo locale.

Nel caso fosse verificata la situazione 1, il minimo locale puntato dall'algoritmo corrisponderebbe non al il minimo locale ideale, cioè quello con periodo più vicino al periodo della soluzione, ma ad uno dei minimi locali interposti tra la soluzione di partenza e la soluzione ideale.

Questo lo si può osservare per esempio nella sperimentazione di tabella 9, dove partendo da un periodo di partenza di 2500s, l'algoritmo convergerebbe il periodo di partenza al minimo locale con periodo in un intorno di 1340s prima di raggiungere il minimo locale ideale.

Questo problema è presente poiché per come è definito l'algoritmo, l'obiettivo non corrisponde all'individuazione del maggior numero di minimi locali, ma la semplice convergenza verso un minimo locale che potenzialmente possa essere il minimo locale ideale.

Per ovviare a queste problematiche ho definito un algoritmo 2 che nel momento nel quale si individua una soluzione dal periodo in un intorno del periodo di partenza restituisca una condizione di eccezione. Questo forzerà l'algoritmo di controllo 3 a

verificare e confrontare gli errori delle soluzioni individuate a partire da una soluzione con periodo a metà della parte superiore e periodo a metà della parte inferiore rispetto al periodo di partenza che ha causato la condizione di eccezione, per determinare il prossimo periodo di partenza.

Per la determinazione della parte superiore e della parte inferiore si utilizza il range 0-5000s come definito della sezione 3.2.2.

Effettuando sperimentazioni dell'algoritmo di controllo 3, ho individuato che queste funzioni in maniera ottimale con problemi con un numero di punti limitato, come ad esempio 10 punti, rispetto a problemi con un numero di punti maggiore come ad esempio 20 o 30 punti. Lo si può notare dalle regioni paretiane delle sperimentazioni (5), (6) e (7), dove nel problema da 10 punti si individuano soluzioni ideali a differenza delle soluzioni per problemi da 30 punti, 20 punti.

Questo è probabilmente spiegato dalla differenza nel numero di vincoli e quindi dalla complessità del problema generato, con il potenziale aumento di minimi locali, e siccome al verificarsi della situazione 2 se distanti dal periodo ideale, l'errore non guida la decisione di una direzione rispetto ad un'altra.

È possibile utilizzare quindi alternativamente in base al numero di punti da interpolare l'algoritmo di controllo 1, che inizializza il problema con una soluzione allo stesso periodo del minimo periodo β e l'algoritmo di controllo 3.

Algoritmo 2: Algoritmo di selezione del periodo di partenza

```

input : sol soluzione corrente individuata
         periodoPartenza periodo di partenza nell'iterazione precedente
output: nuovoPeriodoPartenza periodo di partenza per l'iterazione successiva
1 if sol[periodo] = periodoPartenza then
2   | return eccezione
3 end
4 if sol[periodo] > periodoPartenza then
5   | nuovoPeriodoPartenza  $\leftarrow \frac{5000 - \text{periodoPartenza}}{2} + \text{periodoPartenza}$ 
6 else
7   | nuovoPeriodoPartenza  $\leftarrow \frac{\text{periodoPartenza}}{2}$ 
8 end
9 return nuovoPeriodoPartenza

```

Algoritmo 3: Algoritmo di controllo del modello

input : pt Collezione di punti da interpolare $[(t_1, e_1), \dots, (t_k, e_k)]$
output: soluzioni Insieme di soluzioni che interpolano i punti pt

```

1 passo  $\leftarrow 1$ 
2  $\beta \leftarrow periodoIniziale$ 
3  $periodoPartenza \leftarrow periodoPartenzaIniziale$ 
4 soluzioni  $\leftarrow \emptyset$ 
5 while  $\beta \leq periodoLimite$  do
6   |  $solPrecedente \leftarrow sol$ 
7   |  $sol \leftarrow Interpolare(pt, \beta, solutore, periodoPartenza)$ 
8   | Aggiungi  $sol$  a soluzioni
9   | if  $sol[periodo] = solPrecedente[periodo]$  then
10  |   |  $passo = f(passo)$ 
11  | else
12  |   |  $passo \leftarrow 1$ 
13  | end
14  |  $\beta \leftarrow \beta + passo$ 
15  | try:
16  |   |  $periodoPartenza \leftarrow$  chiamata all'algoritmo 2
17  | catch eccezione:
18  |   | Chiamata all'algoritmo 4 per la gestione dell'eccezione
19  | end
20 end

```

Algoritmo 4: Algoritmo per la gestione dell'eccezione

```

1  $periodoSuperiore \leftarrow \frac{5000 - periodoPartenza}{2} + periodoPartenza$ 
2  $periodoInferiore \leftarrow \frac{periodoPartenza}{2}$ 
3  $soluzioneSuperiore \leftarrow Interpolare(pt, \beta, solutore, periodoSuperiore)$ 
4  $soluzioneInferiore \leftarrow Interpolare(pt, \beta, solutore, periodoInferiore)$ 
5 Aggiungi  $soluzioneSuperiore$  a soluzioni
6 Aggiungi  $soluzioneInferiore$  a soluzioni
7 if  $soluzioneSuperiore[errore] < soluzioneInferiore[errore]$  then
8   |  $periodoPartenza \leftarrow$  chiamata all'algoritmo 2 con  $periodoSuperiore$  e
   |  $periodoPartenza$ 
9 else
10  |  $periodoPartenza \leftarrow$  chiamata all'algoritmo 2 con  $periodoSuperiore$  e
   |  $periodoPartenza$ 
11 end

```

I parametri per l'algoritmo 3 utilizzati per le sperimentazioni sono:

- $periodoIniziale = 0s$
- $periodoLimite = 5000s$
- $periodoPartenzaIniziale = 4000s$

Tabella 10: Soluzioni individuate con minor scostamento dal periodo del problema per ogni sperimentazione

Periodo del problema (s)	parametri della soluzione	10 punti	20 punti	30 punti
7.85	errore (m^2)	2.2e-10	0.47	0.49
	periodo (s)	7.85	5214.7	2235.6
502.4	errore (m^2)	8.6e-12	0.001	0.0
	periodo (s)	516.3	1404.1	499.9
4830.7	errore (m^2)	6.3e-10	0.0	0.0
	periodo (s)	4868.9	4437.5	4750.0

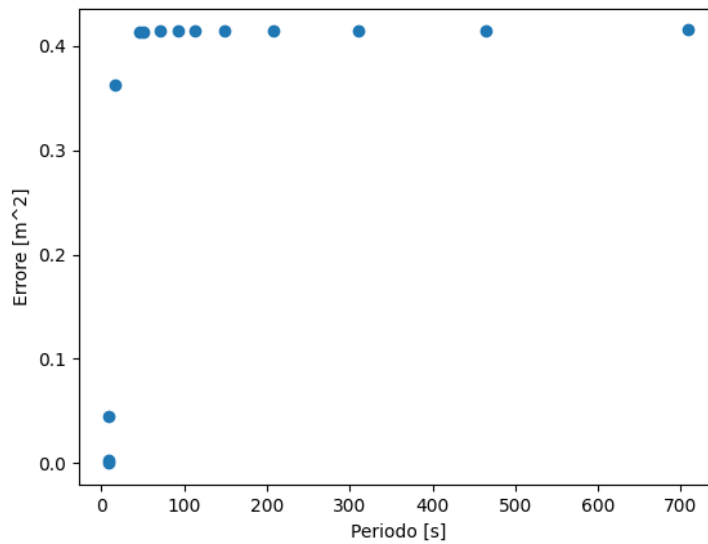


Figura 5: Regione paretiana per problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.8 \text{ rad/s}$, periodo di 7.85s

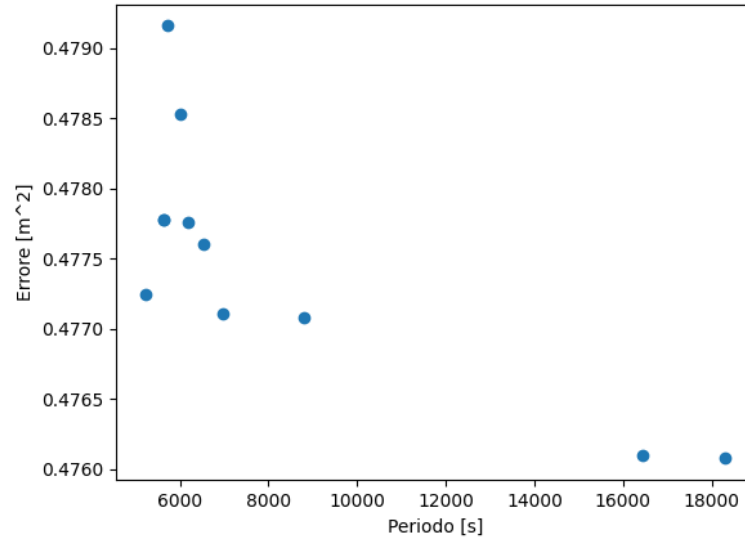


Figura 6: Regione paretiana per problema di 20 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.8 \text{ rad/s}$, periodo di 7.85s

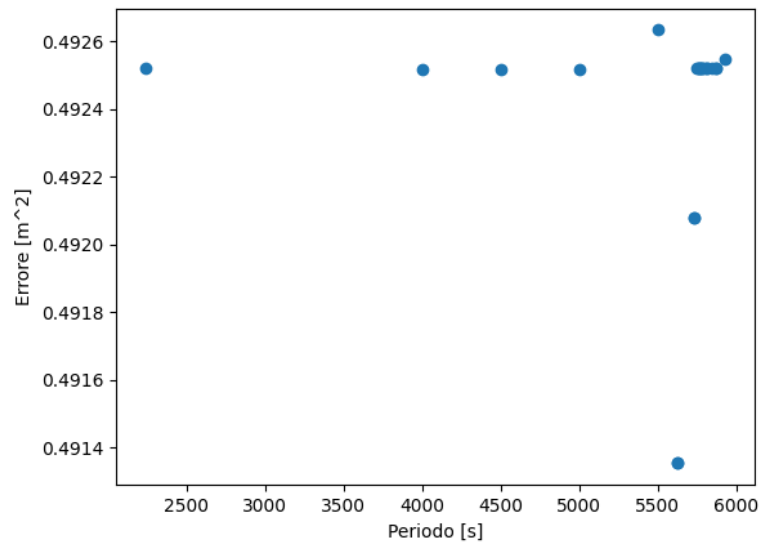


Figura 7: Regione paretiana per problema di 30 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.8 \text{ rad/s}$, periodo di 7.85s

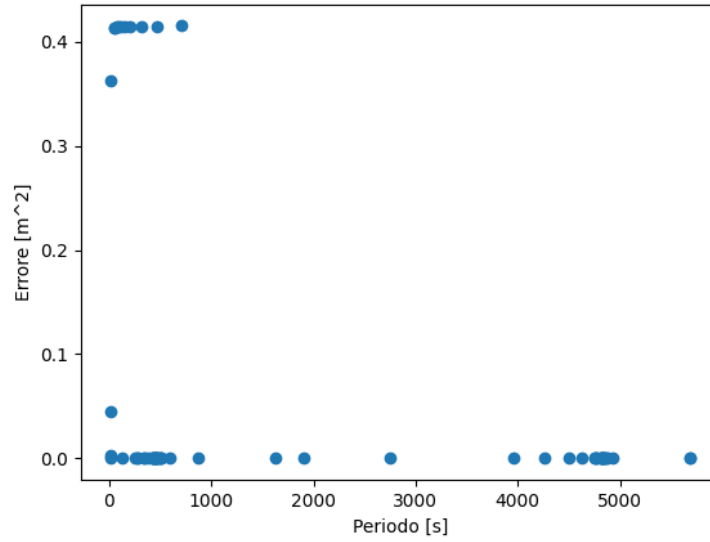


Figura 8: Regione paretiana per problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.0125 \text{ rad/s}$, periodo di 502.4s

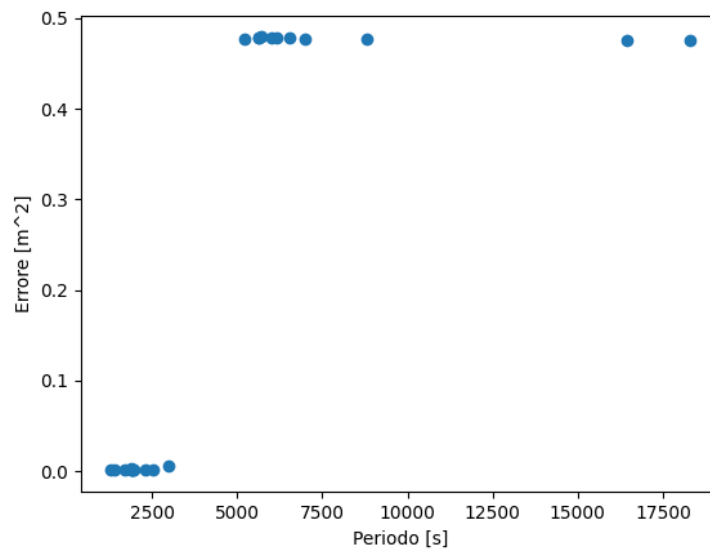


Figura 9: Regione paretiana per problema di 20 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.0125 \text{ rad/s}$, periodo di 502.4s

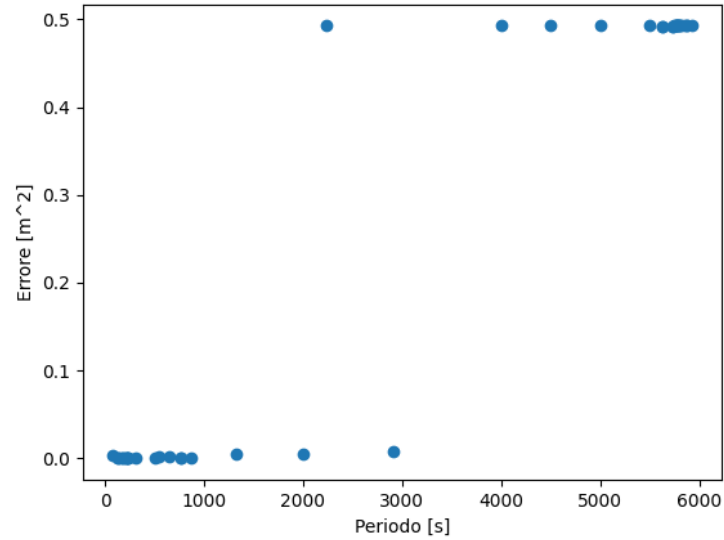


Figura 10: Regione paretiana per problema di 30 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.0125 \text{ rad/s}$, periodo di 502.4s

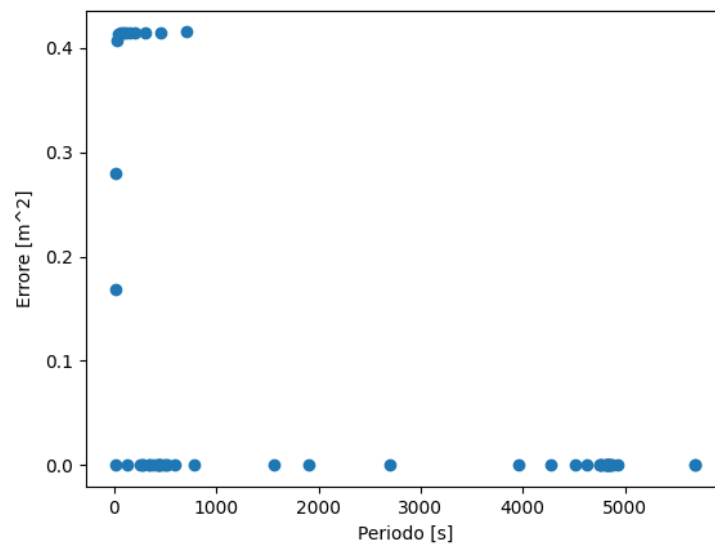


Figura 11: Regione paretiana per problema di 10 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.0013 \text{ rad/s}$, periodo di 4830.7s

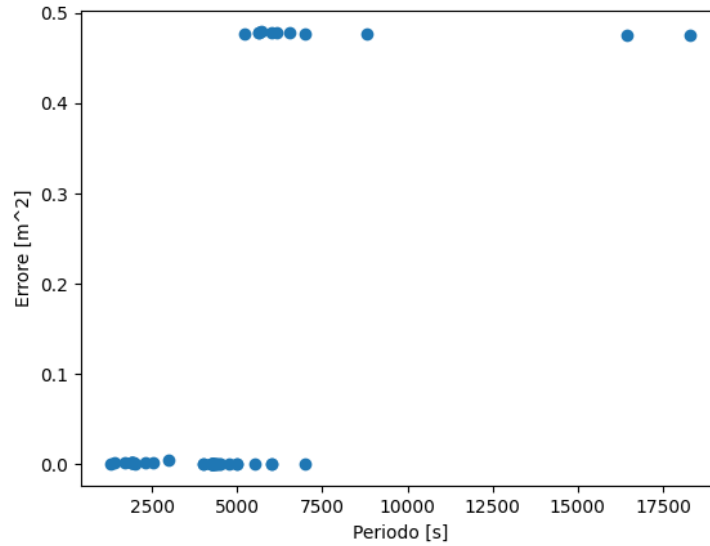


Figura 12: Regione paretiana per problema di 20 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.0013 \text{ rad/s}$, periodo di 4830.7s

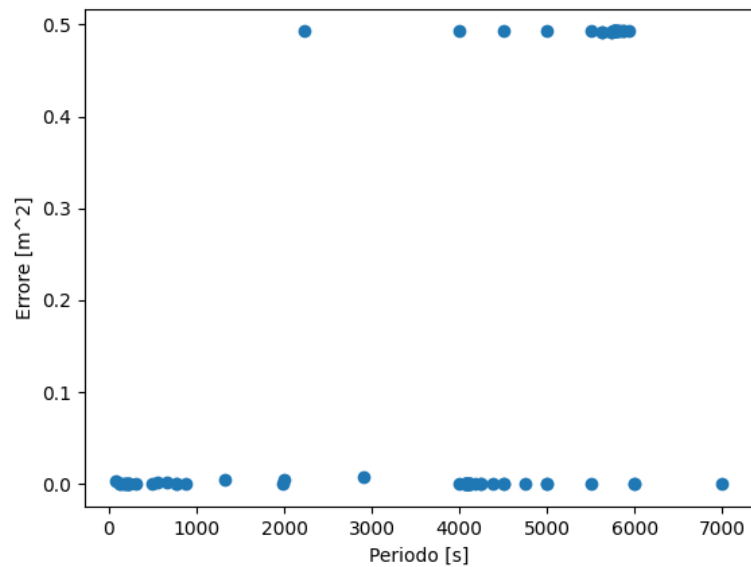


Figura 13: Regione paretiana per problema di 30 punti generati dalla sinusoide $e_i = \sin(\omega t_i)$ con $\omega = 0.0013 \text{ rad/s}$, periodo di 4830.7s

3.2.4 Scelta del solutore

A causa dei vincoli di utilizzo dei solutori globali in `nlopt` [1], la scelta del solutore è ricaduta ai solutori locali (riga 6). È possibile condurre la selezione dell'algoritmo solutore valutandoli su problemi di interpolazione definiti da punti appartenenti ad una stessa sinusoide $e(t) = \sin(\omega t)$ per un numero di punti definito. Le ascisse dei n punti assumono i valori interi dell'insieme $[0, n]$. Analizzo il numero totale di soluzioni individuate, il numero di soluzioni individuate in un intorno del periodo della sinusoide del problema, infine il tempo di calcolo. L'intorno scelto è di dimensioni pari a 100s, è possibile scegliere un diverso intorno in base alle proprie necessità di precisione rispetto alla sinusoide da individuare.

I risultati delle sperimentazioni sono illustrati nelle tabelle da 11 a 15. I parametri dell'algoritmo 1 per le seguenti sperimentazioni sono:

- $f(x)$ in riga 9, algoritmo 1 pari alla funzione (9) con $k = 2$
- $periodoIniziale = 0s$
- $periodoLimite = 5000s$

Tabella 11: Prestazioni dei solutori: Sinusoide con $\omega = 0.8 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
COBYLA	120	21	6.1
BOBYQA	19	6	0.9
NEWUOA	113	101	5.9
PRAXIS	139	139	6.9
SBPLX	26	9	1.3

Tabella 12: Prestazioni dei solutori: Sinusoide con $\omega = 0.0125 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
COBYLA	141	19	7.1
BOBYQA	20	0	1.0
NEWUOA	114	0	5.7
PRAXIS	139	0	6.9
SBPLX	26	1	1.3

Tabella 13: Prestazioni dei solutori: Sinusoide con $\omega = 0.005 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
COBYLA	218	3	11.0
BOBYQA	20	0	1.0
NEWUOA	113	0	5.6
PRAXIS	149	0	7.4
SBPLX	25	0	1.2

Tabella 14: Prestazioni dei solutori: Sinusoide con $\omega = 0.0013 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
COBYLA	193	4	9.8
BOBYQA	19	0	0.9
NEWUOA	113	0	5.6
PRAXIS	146	0	7.3
SBPLX	24	0	1.2

Tabella 15: Prestazioni dei solutori: Sinusoide con $\omega = 0.0010 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
COBYLA	54	0	2.7
BOBYQA	19	0	0.9
NEWUOA	113	0	5.6
PRAXIS	144	0	7.2
SBPLX	27	0	1.3

È interessante analizzare il comportamento dei solutori se il modello utilizzasse trasformasse in vincolo la funzione dell'errore (funzione 4) invece del periodo: l'algoritmo scelto dall'insieme disponibile è COBYLA [2], da come si può notare, è l'unico solutore che individua soluzioni nell'intorno specificato. È da evidenziare la difficoltà

per il solutore COBYLA nel individuare soluzioni al di fuori del range di esplorazione fissato per il criterio d'arresto, come da esempio in tabella 15 dove i punti del problema sono generati da una sinusoide con periodo pari a $T = \frac{2\pi}{\omega} = \frac{2\pi}{0.0010} = 6280s$.

3.2.5 Gestione degli errori

L'unico errore che può verificarsi da tenere in considerazione è `nlopt.RoundoffLimited`. È un evento che specifica una situazione di errore inerente progressivo, cioè l'errore che si commette rappresentando un numero reale con un numero finito di cifre, intrinseco nei calcolatori. Si può verificare quando si esegue l'implementazione del modello per l'interpolazione dei punti in riga 6 dell'algoritmo 1.

Quando si verifica un eccezione di errore inerente, è possibile rieseguire il solutore aumentando i valori di tolleranza sulla precisione della variazione delle variabili e della funzione obiettivo (riga 18, algoritmo 5).

La tolleranza determina il criterio di arresto nel processo di individuazione di un ottimo locale [1], quando la variazione del valore della funzione obiettivo o delle variabili in ogni direzione è minore della tolleranza impostata, il solutore restituirà l'ultima soluzione individuata, perciò aumentando i valori di tolleranza si impedisce la generazione di un errore inerente progressivo, non considerando a priori, valori dalla precisione elevata.

Algoritmo 5: Algoritmo di controllo del modello con gestione degli errori

input : pt Collezione di punti da interpolare $[(t_1, e_1), \dots, (t_k, e_k)]$
output: soluzioni Insieme di soluzioni che interpolano i punti pt

```

1 passo  $\leftarrow 1$ 
2  $\beta \leftarrow valoreIniziale$ 
3  $tolFun \leftarrow tolleranzaDefault$ 
4  $tolVar \leftarrow tolleranzaDefault$ 
5 soluzioni  $\leftarrow \emptyset$ 
6 while  $\beta \leq periodoLimite$  do
7   try:
8     solPrecedente  $\leftarrow sol$ 
9     sol  $\leftarrow Interpola(pt, \beta, solutore, tolFun, tolVar)$ 
10    Aggiungi sol a soluzioni
11    if sol[periodo] = solPrecedente[periodo] then
12      | passo  $\leftarrow f(passo)$ 
13    else
14      | passo  $\leftarrow 1$ 
15    end
16     $\beta \leftarrow \beta + passo$ 
17  catch RoundoffLimited:
18    | Aumenta tolFun e tolVar
19  end
20 end

```

3.2.6 Selezione della soluzione

La selezione di una soluzione tra le tante individuate determina un passo fondamentale per tutto il modulo del sottoproblema di interpolazione, poiché si potrebbe incorrere nel rischio di scartare la sinusoide che rappresenta esattamente uno dei k satelliti. I due metodi identificati per questo compito si basano su diverse ipotesi sulla sinusoide ideale e sulle caratteristiche di tutte le sinusoidi individuate, questi sono il criterio del punto di utopia, e criterio degli standard solo sull'errore.

Criterio del punto di utopia

Il punto di utopia è la soluzione che nello spazio degli obiettivi ha come coordinate i valori ottimi di ciascuno. Il problema principale che sorge nello applicare questo

criterio al problema di interpolazione, è che non si conosce il valore ottimo ideale dell'obiettivo rispetto al periodo, mentre per l'errore, il valore è pari a 0.

Questo succede poiché non esiste un limite superiore al periodo che specifichi l'insieme di sinusoidi che violino uno dei vincoli del modello, essi avranno unicamente come conseguenza, se non relazionati ai punti, un alto errore.

L'idea che quindi ho elaborato è quella di determinare il punto di utopia in base ai valori dei periodi delle sinusoidi individuate. La coordinata rispetto all'errore viene fissata a 0, mentre per la coordinata rispettiva al periodo, calcolo la media dei periodi delle sinusoidi. Questo si basa sull'ipotesi che se la maggior parte dei periodi, individuati dal solutore, cade in un intorno di un determinato valore, allora tale valore è il periodo della sinusoide ricercata. Determino la soluzione da selezionare scegliendo la soluzione più vicina, valutando la distanza euclidea di ogni soluzione rispetto al punto di utopia. Illustro esempi di generazione e scelta di una soluzione:

I problemi scelti per la valutazione e analisi del comportamento del criterio del punto di utopia, sono problemi definiti per 10, 20, 30 punti generati da una sinusoide con un determinato periodo. I parametri dell'algoritmo 5 per le seguenti sperimentazioni sono:

- $f(x)$ in riga 9, algoritmo 1 pari alla funzione (9) con $k = 2$
- $periodoIniziale = 0s$
- $periodoFinale = 5000s$
- $tolleranzaDefault = 1e-6$
- $e(t) = \sin(0.0010t)$ sinusoide con periodo pari a: $T = \frac{2\pi}{\omega} = \frac{2\pi}{0.0010} = 6280s$

Tabella 16: periodo da individuare uguale a 6280s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	662	4.5	2.7e-8
20	499	5.4	3.6e-5
30	1342	3.09	1.7e-4

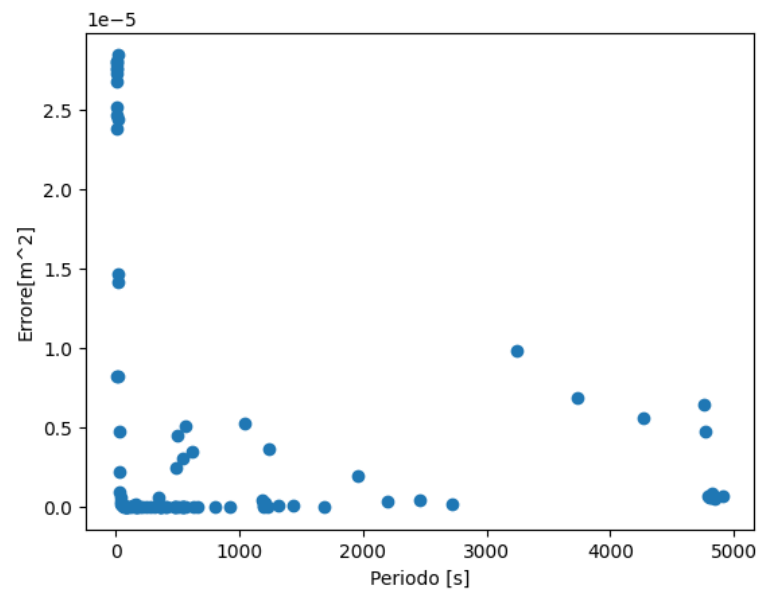


Figura 14: Regione paretiana per problema da 10 punti di tabella 16

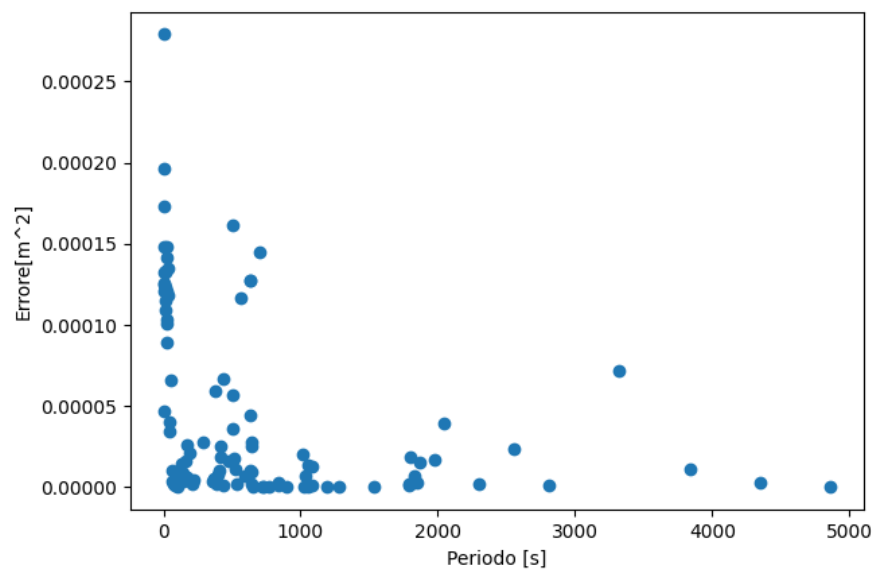


Figura 15: Regione paretiana per problema da 20 punti di tabella 16

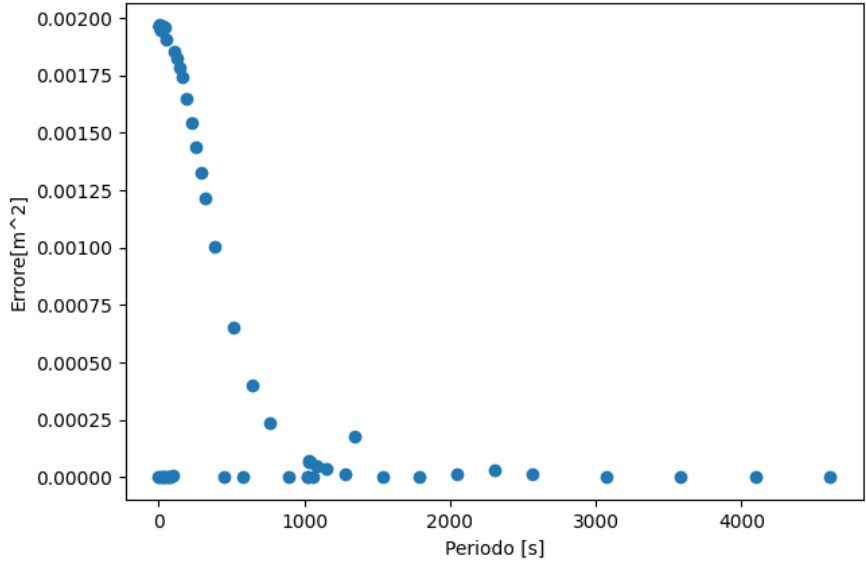


Figura 16: Regione paretiana per problema da 30 punti di tabella 16

- $e(t) = \sin(0.0013t)$ sinusoide con periodo pari a: $T = 4830.7s$

Tabella 17: periodo da individuare uguale a 4830.7s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	1201	7.5	1.3e−11
20	2095	10.3	7.5e−5
30	4175	10.9	3.2e−4

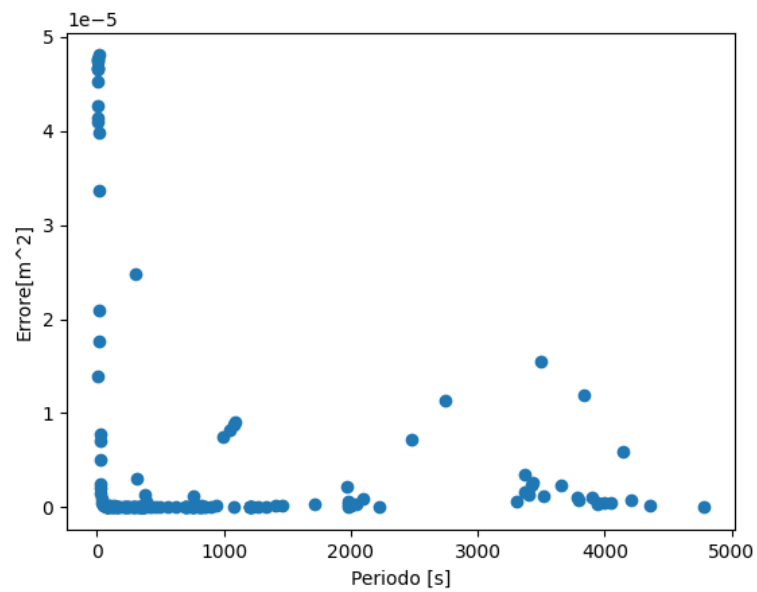


Figura 17: Regione paretiana per problema da 10 punti di tabella 17

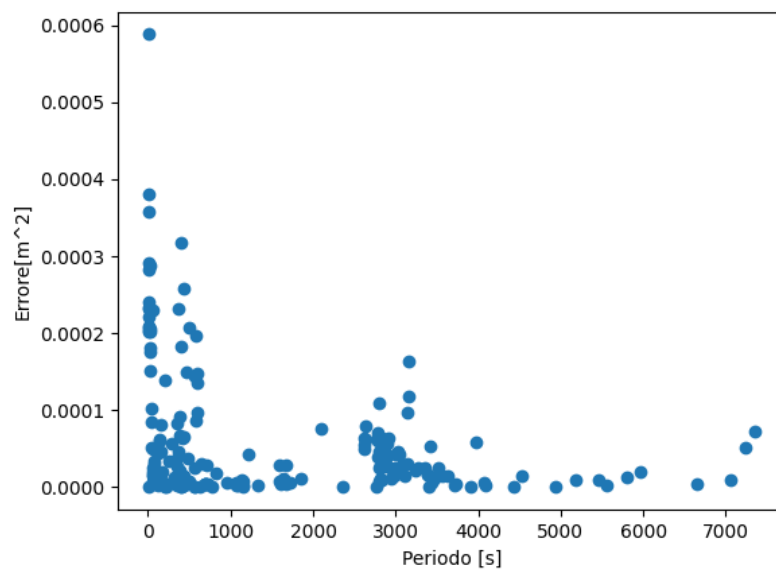


Figura 18: Regione paretiana per problema da 20 punti di tabella 17

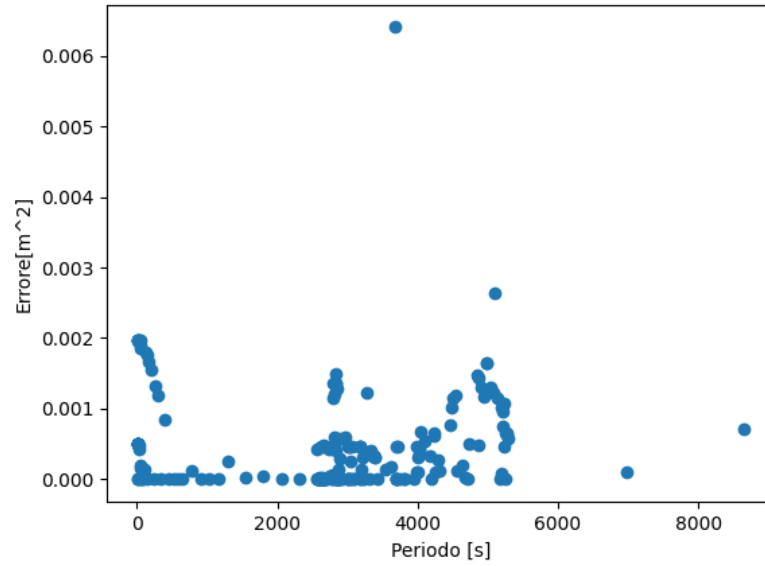


Figura 19: Regione paretiana per problema da 30 punti di tabella 17

- $e(t) = \sin(0.005t)$ sinusoide con periodo pari a: $T = 1256s$

Tabella 18: periodo da individuare uguale a 1256s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	753	6.8	$1.3e-6$
20	1321	16.2	$1.7e-4$
30	1681	8.2	$9.6e-4$

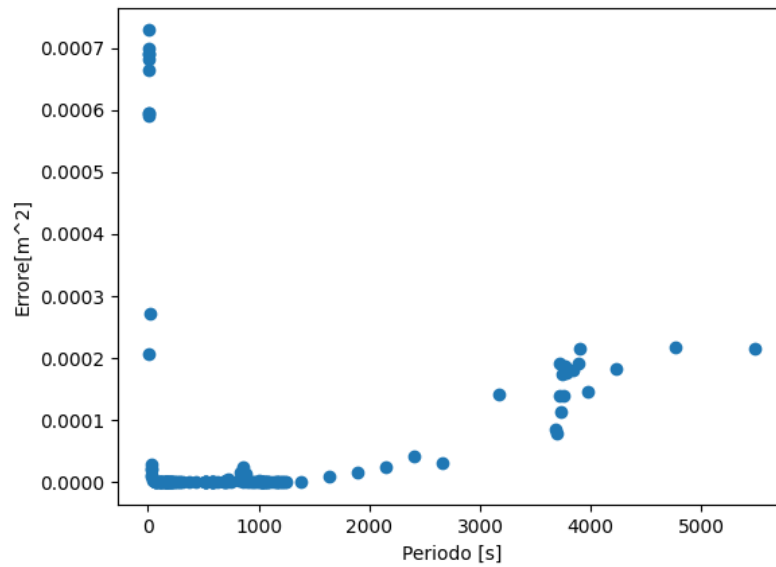


Figura 20: Regione paretiana per problema da 10 punti di tabella 18

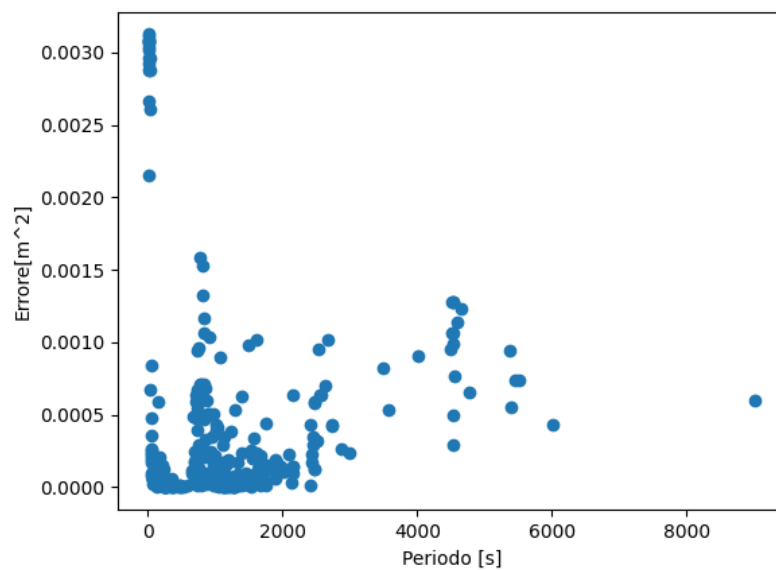


Figura 21: Regione paretiana per problema da 20 punti di tabella 18

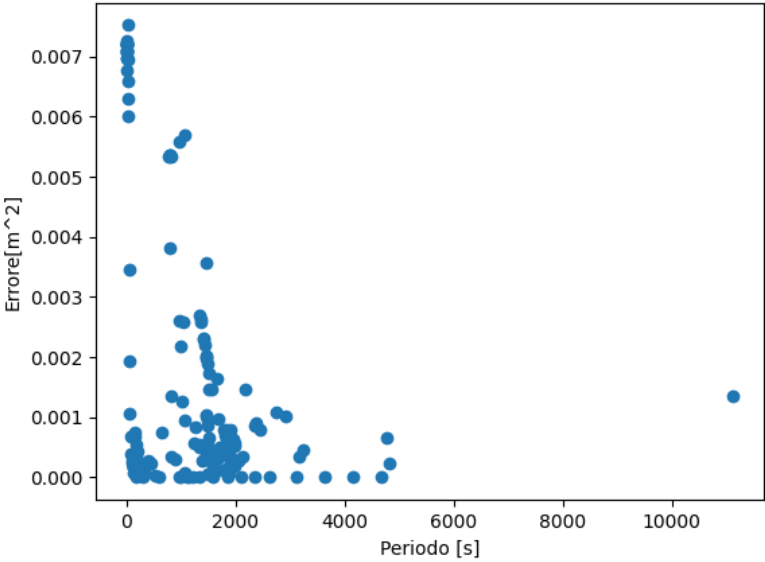


Figura 22: Regione paretiana per problema da 30 punti di tabella 18

- $e(t) = \sin(0.0125t)$ sinusoide con periodo pari a: $T = 502.4s$

Tabella 19: periodo da individuare uguale a 502.4s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	425	6.4	$2.5e-10$
20	701	18.1	$2.7e-4$
30	589	9.3	$6.0e-4$

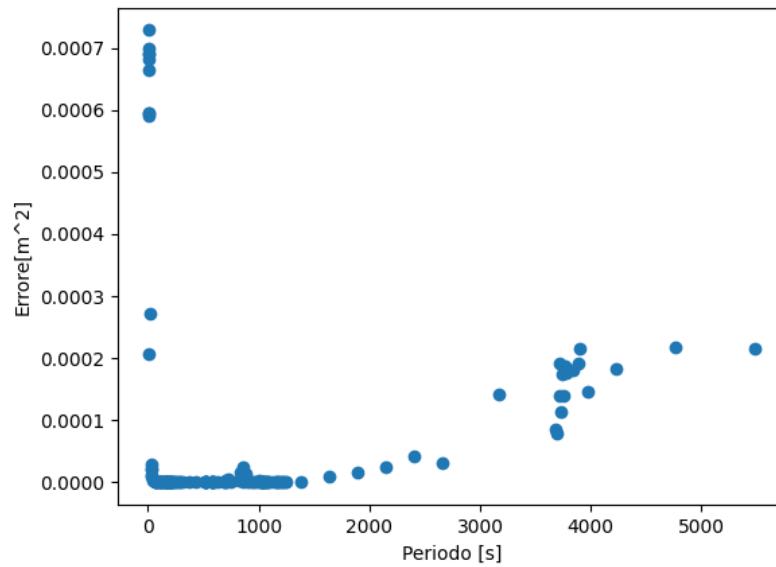


Figura 23: Regione paretiana per problema da 10 punti di tabella 19

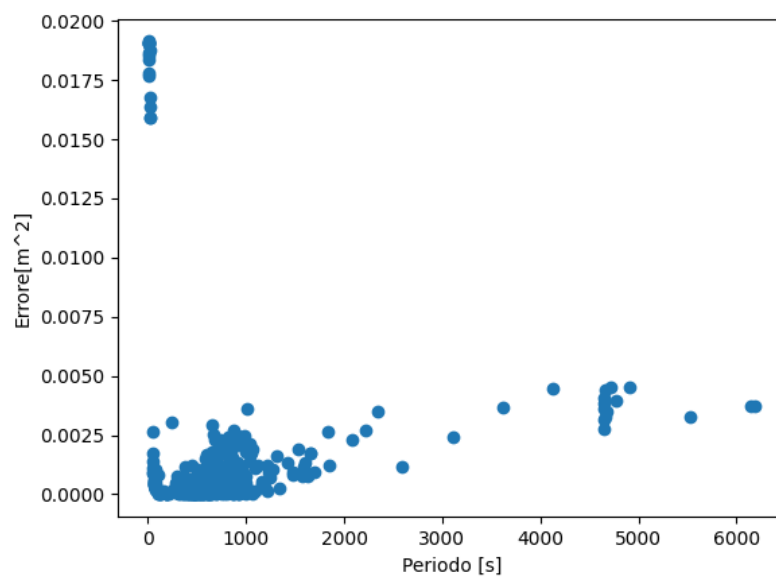


Figura 24: Regione paretiana per problema da 20 punti di tabella 19

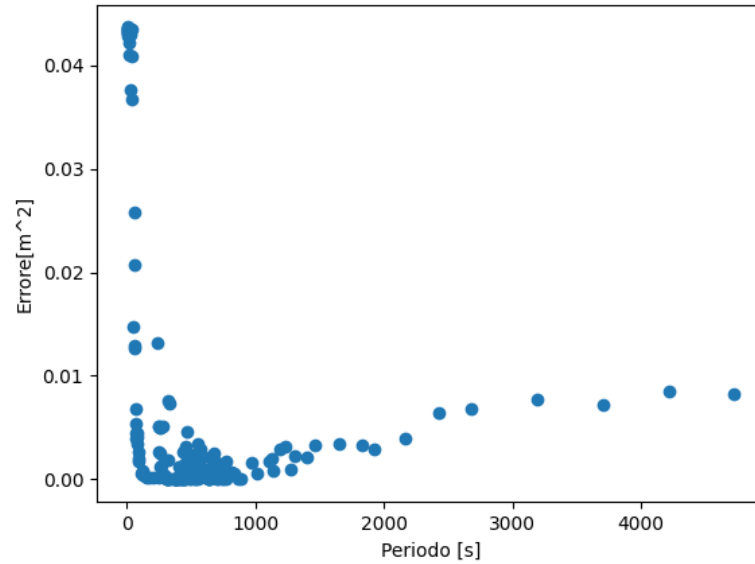


Figura 25: Regione paretiana per problema da 30 punti di tabella 19

- $e(t) = \sin(0.8t)$ sinusoide con periodo pari a: $T = 7.85s$

Tabella 20: periodo da individuare uguale a 7.85s






Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	40	2.2	$1.3e-6$
20	8313	97.0	$1.7e-4$
30	4519	12.1	$9.6e-4$

L'esecuzione dell'algoritmo è caratterizzato da uno scostamento medio dal periodo da individuare di $2377s$ e una mediana di $1278s$ ed infine un tempo medio di esecuzione di $13.62s$ e una mediana di $6.8s$.

La mediana e la media dello scostamento denotano un alta differenza tra la soluzione scelta dall'algoritmo e la sinusoide del problema, la causa di ciò è da individuarsi nel posizionamento del punto di utopia ed una mancata normalizzazione dei domini dell'errore e periodo.

Prendendo ad esempio la tabella 18, in particolare la soluzione individuata per il problema da 20 punti ed suo il grafico delle soluzioni generate, figura 26 e figura 27, si può notare che una soluzione con scostamento minore è stata individuata, ma non scelta dall'algoritmo di selezione poiché la distanza euclidea dal punto di utopia risulta maggiore rispetto alla soluzione scelta.

Legenda per i grafici da figura 26 a figura 32:

-  : soluzione individuata.
-  : soluzione individuata scelta dal criterio.
-  : punto di utopia.
-  : soluzione individuata ideale.
-  : punto della sinusoide del problema.

Da come si può notare dal primo grafico di figura 26 il punto di utopia risulta essere traslato al di fuori della zona dove la maggior parte delle soluzioni sono situate, a causa di soluzioni outlier.

È necessario quindi implementare un metodo di determinazione del periodo del punto di utopia migliore o passare ad un altro criterio di selezione.

Un miglioramento può essere l'implementazione di una media pesata, dove si determina il peso in base al errore della soluzione, questo permette di dare più rilevanza alle soluzioni con errore minore, comprendendo però in questo modo anche soluzioni con basso periodo rispetto al periodo della sinusoide da individuare.

Una soluzione alternativa può essere la modifica dell'algoritmo di controllo ed esplorazione dei periodi, modificando la gestione del passo. Diminuendo il passo quando si individua una soluzione con errore minore di un determinato valore di soglia, si approfondisce e si aumenta il numero di soluzioni individuate in un intorno di esso.

L'altra probabile causa è la mancata normalizzazione dei domini dell'errore e del periodo. Si possono notare gli effetti di ciò nel grafico in figura 29 corrispondente alla tabella 20 per il problema da 20 punti.

Si può notare, dai grafici in figura 30 e 31 che la distanza tra il punto di utopia e la soluzione con periodo che si scosta meno dalla soluzione da individuare, è approssimativamente pari ad un valore di 8000s, molto maggiore rispetto ad una soluzione che ha lo stesso periodo del punto di utopia ma con uno scostamento maggiore, pari a 9.5

Un altro fattore che ha determinato la mancata scelta della soluzione ideale, mostrata in figura 30, sono il numero di soluzioni individuate in un suo intorno limitato,

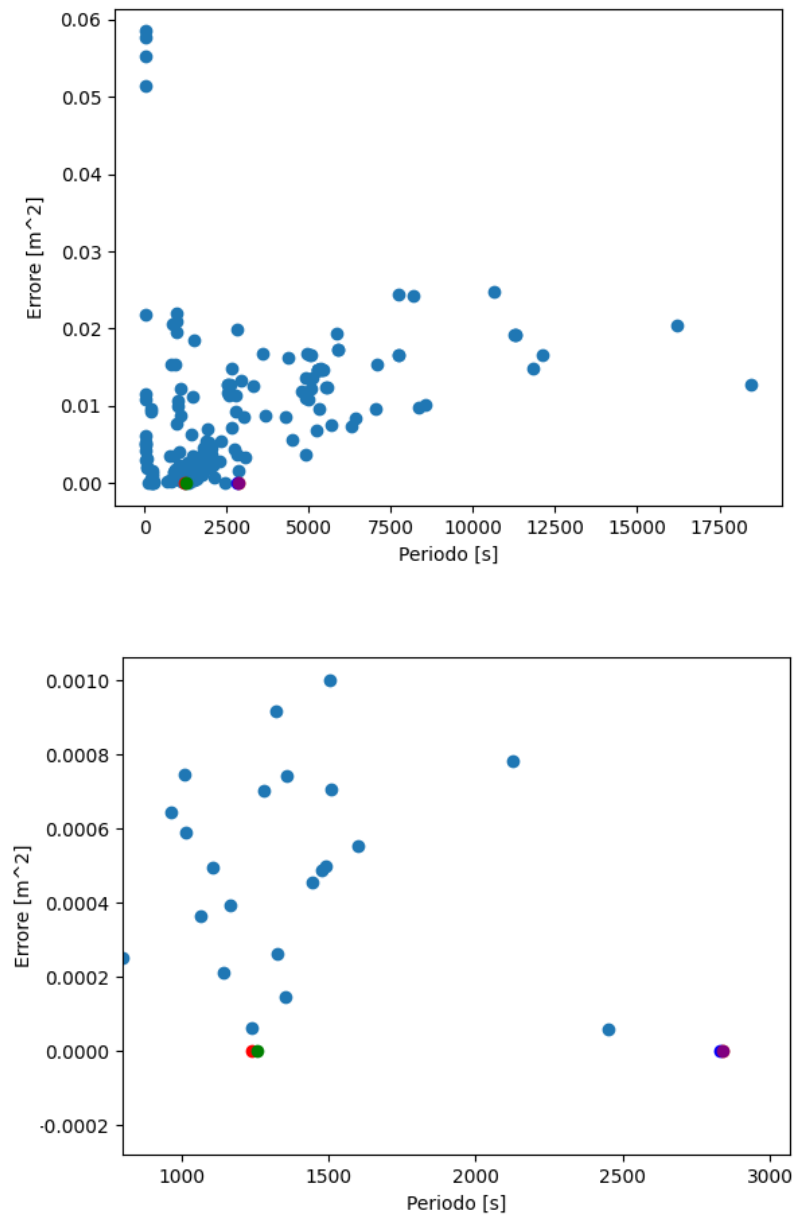


Figura 26: Soluzioni generate per il problema di 18 con 20 punti

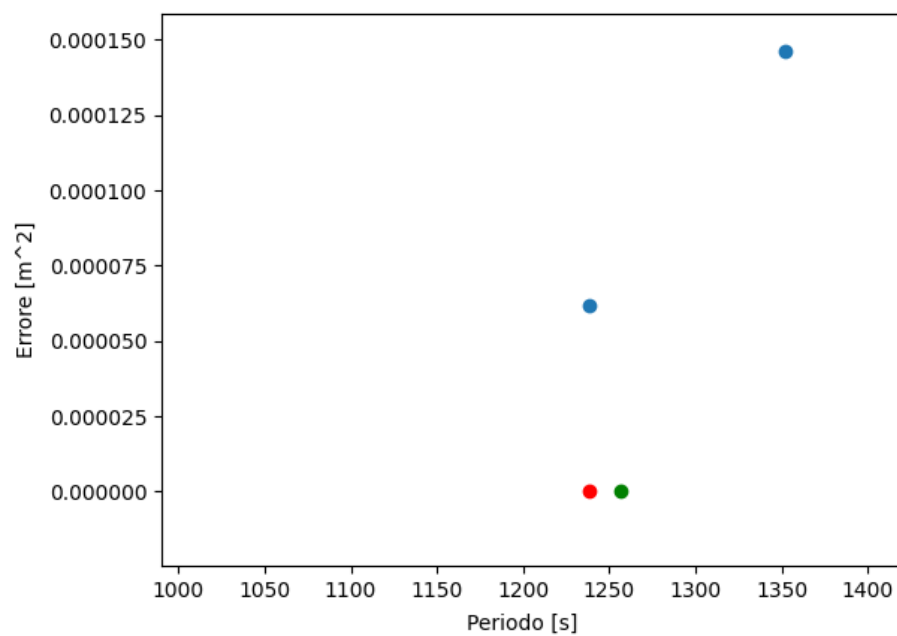


Figura 27: soluzione ideale per il problema di tabella 18 con 20 punti

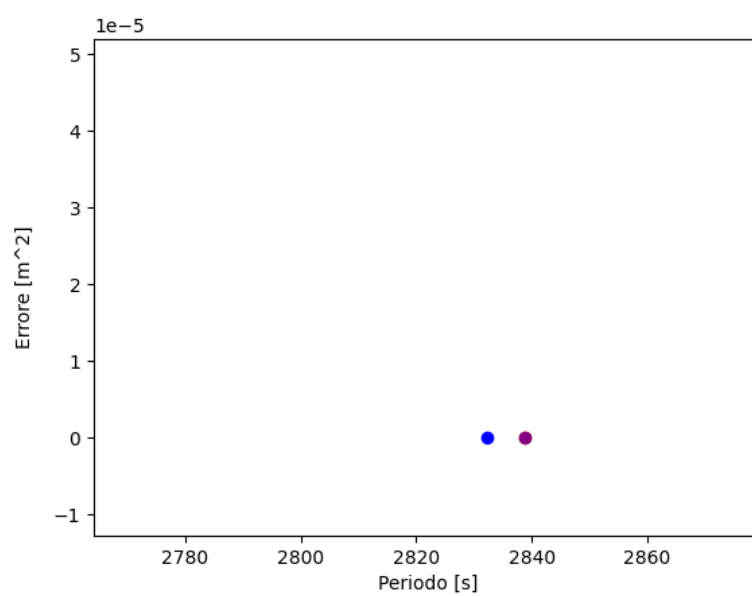


Figura 28: soluzione scelta per il problema di tabella 18 con 20 punti

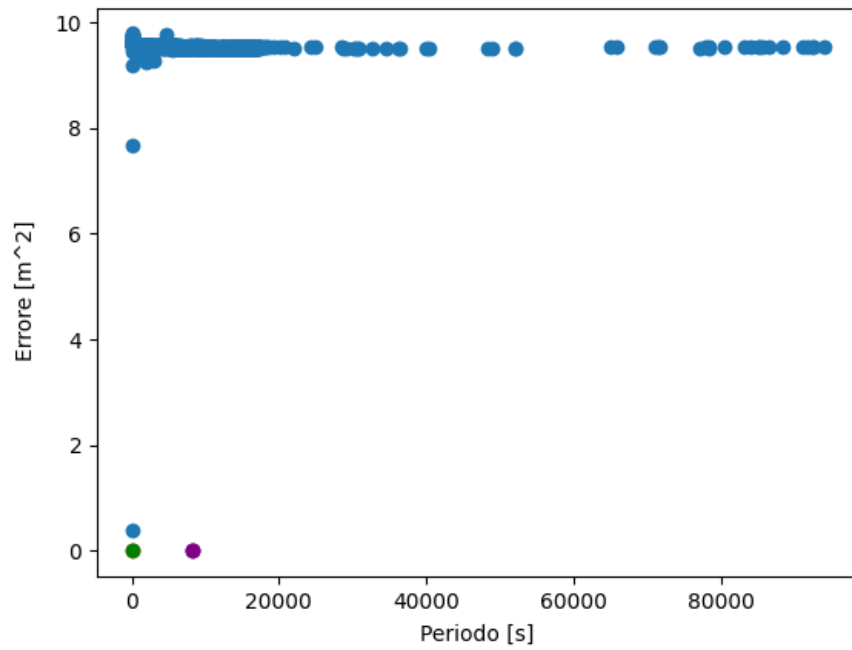


Figura 29: Soluzioni generate per il problema di tabella 20 con 20 punti

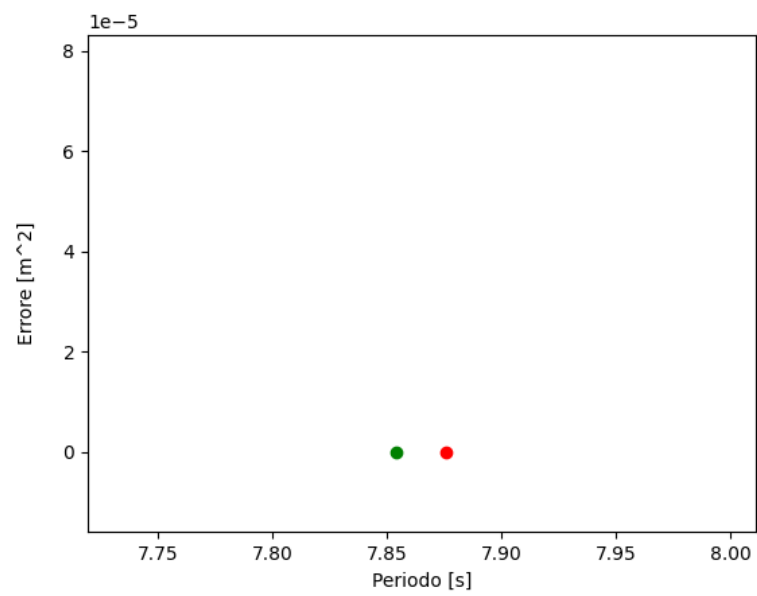


Figura 30: Soluzione ideale per il problema di 20 con 20 punti

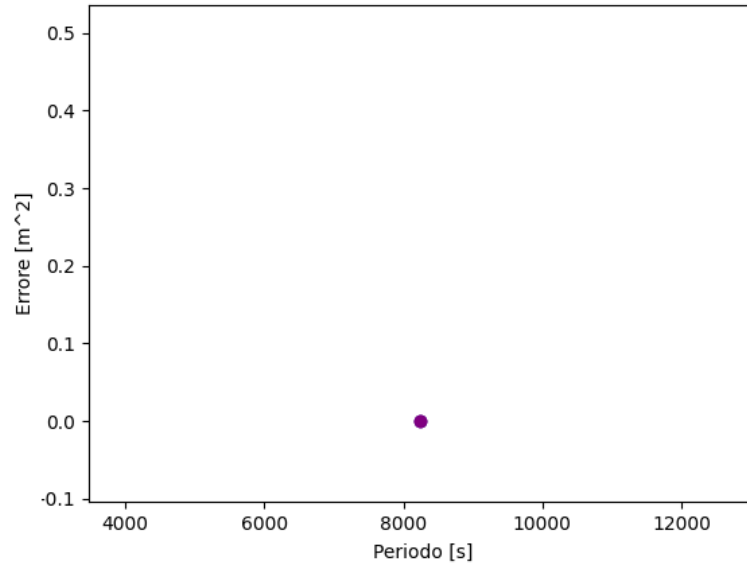


Figura 31: Posizione del punto di utopia per il problema di 20 con 20 punti

come per esempio nell'insieme dei periodi $[0, 20]$ illustrato in figura 32, dove sono individuate solo 6 soluzioni rispetto al numero totale di soluzioni individuate pari a 1527 in questo esempio.

Il problema di questa metodologia di specifica del periodo per il punto di utopia è che è basata in maniera marcata sulle prestazioni, il numero e la tipologia di soluzioni individuate dall'algoritmo di esplorazione dei periodi.

Riassumendo, per il miglioramento del criterio di selezione esplicito le seguenti possibili modifiche:

- La normalizzazione dei domini di errore e di periodo per tutte le soluzioni individuate
- L'implementazione di una media pesata e quindi il calcolo del peso per ogni soluzione individuata in base al suo errore.
- Aggiunta di condizione di diminuzione del passo nel algoritmo esplorazione dei periodi.

Dato le ampie modifiche richieste, ho optato per l'individuazione e definizione di un altro criterio di selezione.

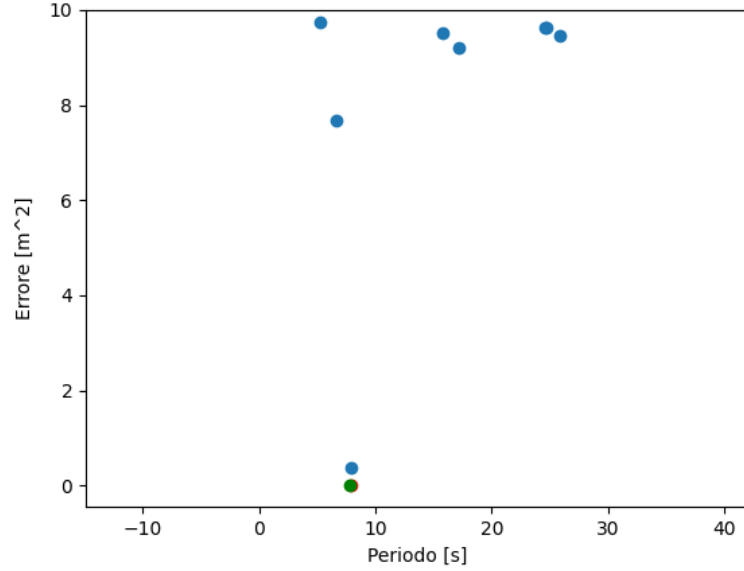


Figura 32: Soluzioni individuate nell'insieme di periodi $[0, 20]$ per il problema di tabella 20 con 20 punti

Criterio degli standard sull'errore

Gli standard sono valori-soglia al di sotto dei quali non si vuole che gli obiettivi possano peggiorare. Data la non predicibilità del periodo ottimale in questo problema, è possibile definire gli standard solo per l'errore. La modalità con il quale ho definito gli standard per la scelta della soluzione prevede la suddivisione in range del dominio dell'errore, ipotizzando che ogni soluzione in uno stesso range sia equivalente, se si considera solo l'errore. Ogni range è identificato con un numero ordinale specificandoli come livelli: primo livello, secondo livello, \dots , n -esimo livello, dove per ordine crescente dei livelli si ha l'aumento del range d'errore. I range devono essere definiti a priori, in maniera attenta rispetto all'errore intrinseco dello strumento di rilevazione dei punti. È necessario inoltre effettuare ipotesi sul errore di una soluzione: errore di una soluzione ottimale, errore di una soluzione non correlata ai punti, errore di una soluzione legata ad un sottoproblema con un assegnamento di punti errato.

Il seguente pseudocodice illustra una procedura di una struttura dati che ho definito, per contenere la soluzione con livello più basso possibile e con più alto periodo tra tutte le soluzioni individuate all'interno del corrispettivo livello, perciò gli standard dell'errore sono fissati dalla soluzione con il livello minore.

Tutte le soluzioni con un livello maggiore della soluzione ottimale vengono scartate (riga 1, algoritmo 6), invece se la soluzione corrente ha un livello inferiore, essa viene

Algoritmo 6: Criterio degli standard

```

input : solCorrente tupla contenente (errore, periodo, ampiezza, fase)
1 if LivelloDi(solCorrente) < LivelloDi(solOttimale) then
2   | solOttimale  $\leftarrow$  solCorrente
3 else
4   | if LivelloDi(solCorrente) = LivelloDi(solOttimale) then
5     | if LivelloDi(solCorrente) = ultimoLv then
6       | if solCorrente[errore] < solOttimale[errore] then
7         | | solOttimale  $\leftarrow$  solCorrente
8       | end
9     | else
10      | if solCorrente[periodo] > solOttimale[periodo] then
11        | | solOttimale  $\leftarrow$  solCorrente
12      | end
13    | end
14  end
15 end

```

memorizzata come soluzione ottimale. Se i livelli della soluzione ottimale e della soluzione corrente si equivalgono, si memorizza la soluzione corrente solo se essa ha un periodo maggiore della soluzione ottimale (riga 10, algoritmo 6).

Solo l'ultimo livello ha un criterio di selezione della soluzione ottimale differente, poiché si seleziona la soluzione con il minimo errore (riga 6, algoritmo 6). L'ultimo livello, rappresenta l'insieme dei valori d'errore più alti, che vanno da un valore fissato a $+\infty$, perciò all'interno di questo insieme illimitato, cerco di selezionare la soluzione che più si correla all'assegnamento dei punti.

Illustro i risultati dell'implementazione ed esecuzione del criterio degli standard, generando le soluzioni utilizzando il modello basato sul metodo dei vincoli, variando il minimo periodo β nel range fissato (sezione 3.2.2).

I livelli fissati sono

- Livello 1: errore < $1e-5$
- Livello 2: errore < $1e-2$
- Livello 3: errore < $1e-1$
- Livello 4: errore < 1
- Livello 5: errore < 3

- Livello 6: errore < 5
- Livello 7: errore < 7
- Livello 8: errore < 10
- Livello 9: errore ≥ 10

Ho fissato i range in maniera da ottenere maggiore precisione e diversificazione delle soluzioni con errore tra 0 e 1, suddividendo tale spazio in 4 livelli. Nella definizione dei livelli, non ho tenuto in considerazione l'errore intrinseco dello strumento di rilevazione, per ottenere delle prime informazioni sulla prestazione dell'algoritmo in una situazione ideale. A seguito si elencano problemi con punti generati da una sinusoide $e(t)$ e nelle tabelle le soluzioni individuate dall'algoritmo.

I parametri dell'algoritmo 5 per le seguenti sperimentazioni sono:

- $f(x)$ in riga 9, algoritmo 1 pari alla funzione (9) con $k = 5$
- $periodoIniziale = 0s$
- $periodoFinale = 5000s$
- $tolleranzaDefault = 1e-6$
- $e(t) = \sin(0.0010t)$ sinusoide con periodo pari a: $T = \frac{2\pi}{\omega} = \frac{2\pi}{0.0010} = 6280s$

Tabella 21: periodo da individuare uguale a 6280s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	4631.8	2.2	9.0e-7
20	5585.4	1.9	1.0e-34
30	4906.0	2.0	6.0e-4

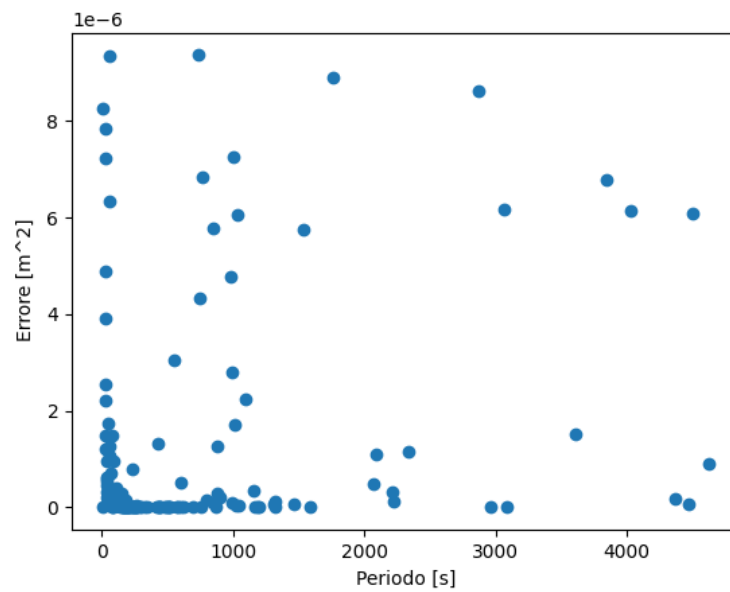


Figura 33: Regione paretiana per problema da 10 punti di tabella 21

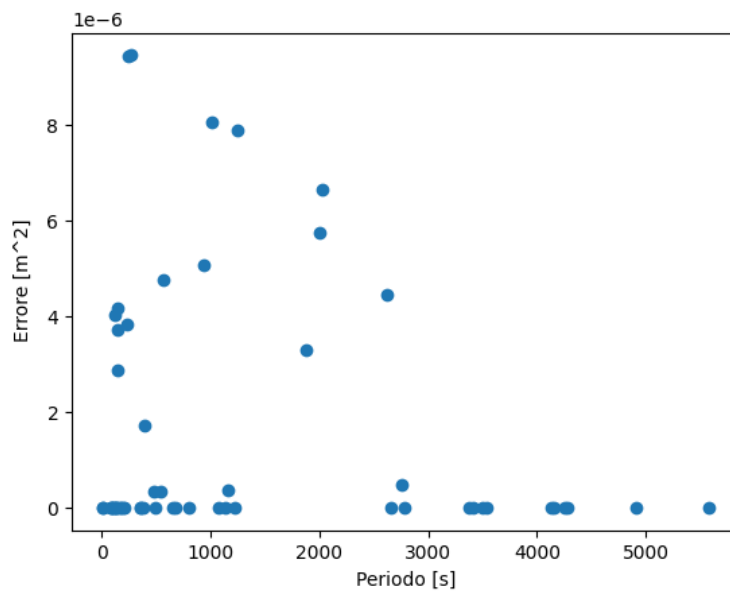


Figura 34: Regione paretiana per problema da 20 punti di tabella 21

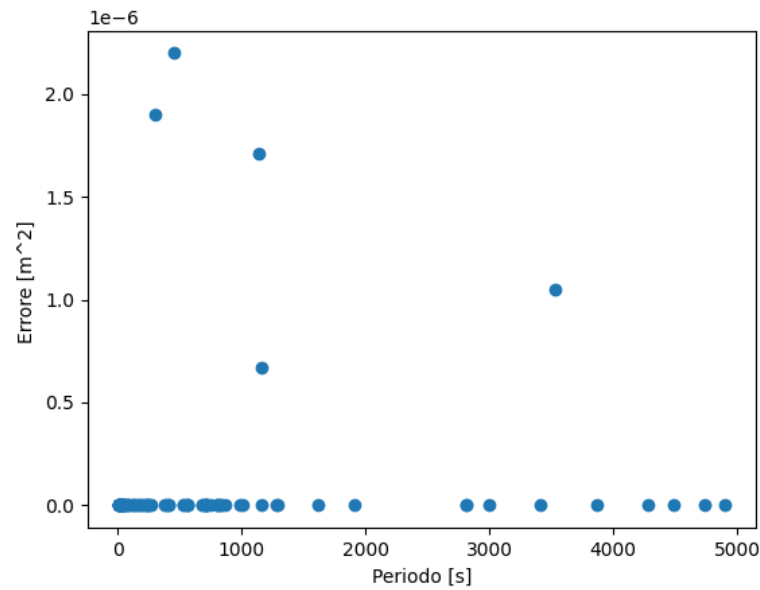


Figura 35: Regione paretiana per problema da 30 punti di tabella 21

- $e(t) = \sin(0.0013t)$ sinusoide con periodo pari a: $T = 4830.7s$

Tabella 22: periodo da individuare uguale a 4830.7s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	4471.5	2.2	6.4e−8
20	4907.0	2.0	0.0
30	4492.0	2.3	0.0

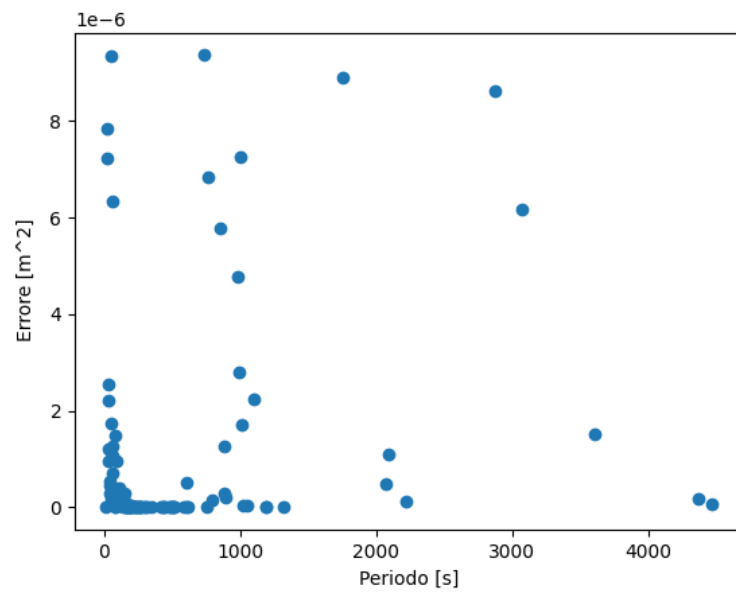


Figura 36: Regione paretiana per problema da 10 punti di tabella 22

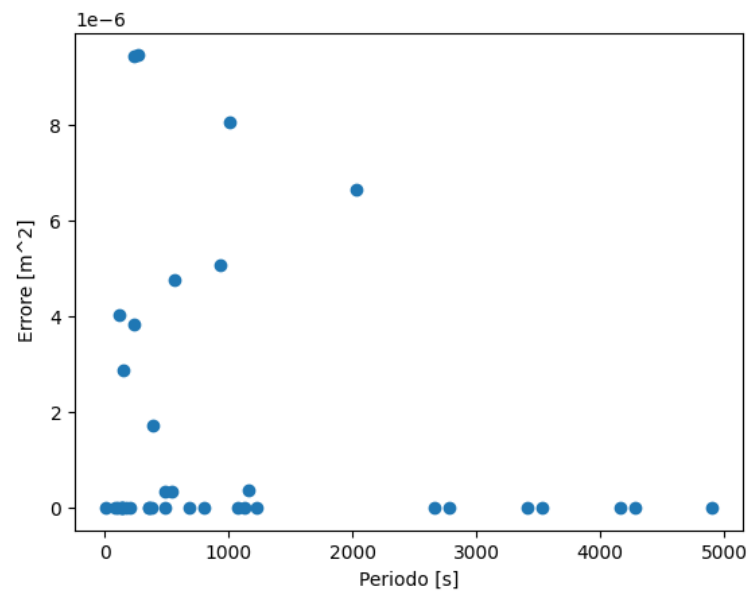


Figura 37: Regione paretiana per problema da 20 punti di tabella 22

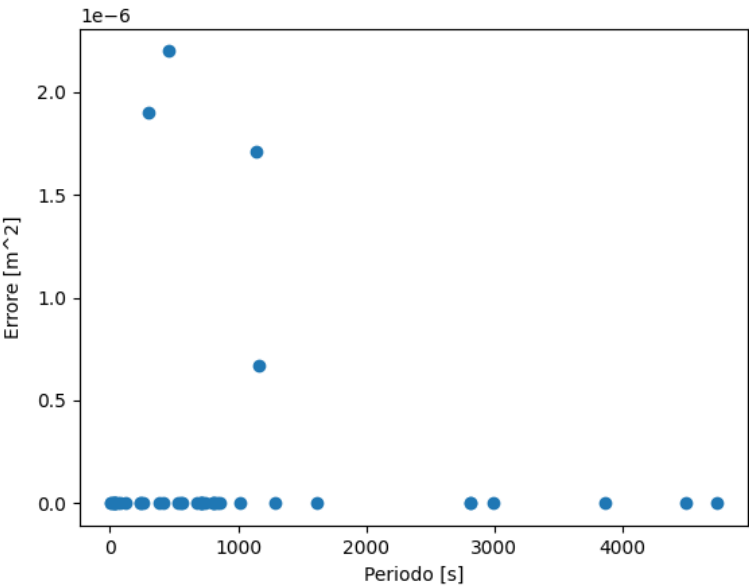


Figura 38: Regione paretiana per problema da 30 punti di tabella 22

- $e(t) = \sin(0.005t)$ sinusoide con periodo pari a: $T = 1256s$

Tabella 23: periodo da individuare uguale a 1256s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	1759.0	2.1	8.9e−6
20	1224.0	2.6	0.0
30	4740	2.0	0.0

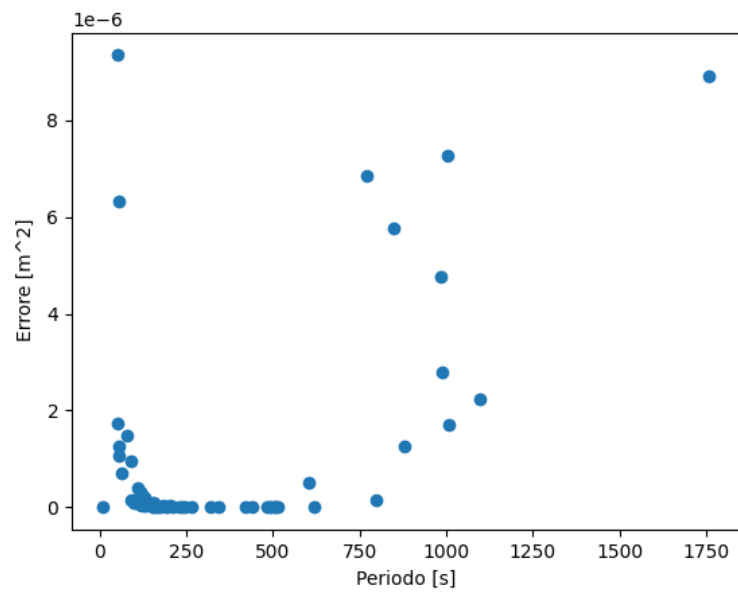


Figura 39: Regione paretiana per problema da 10 punti di tabella 23

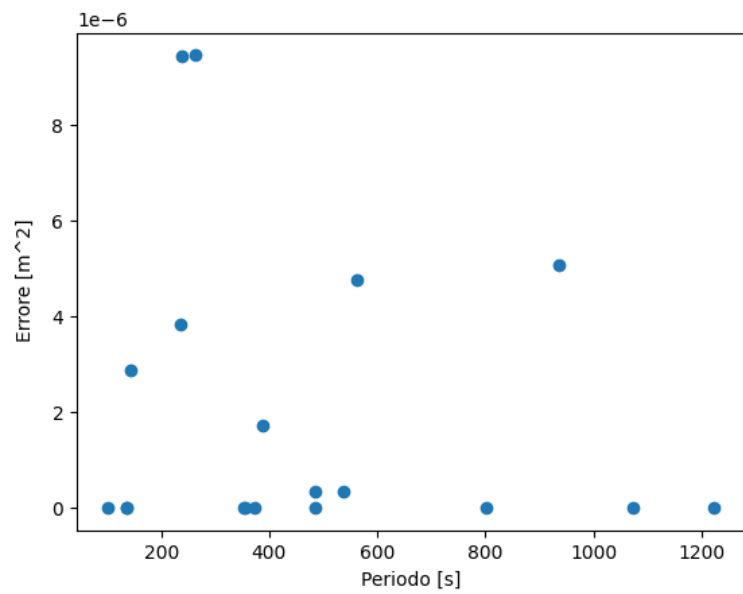


Figura 40: Regione paretiana per problema da 20 punti di tabella 23

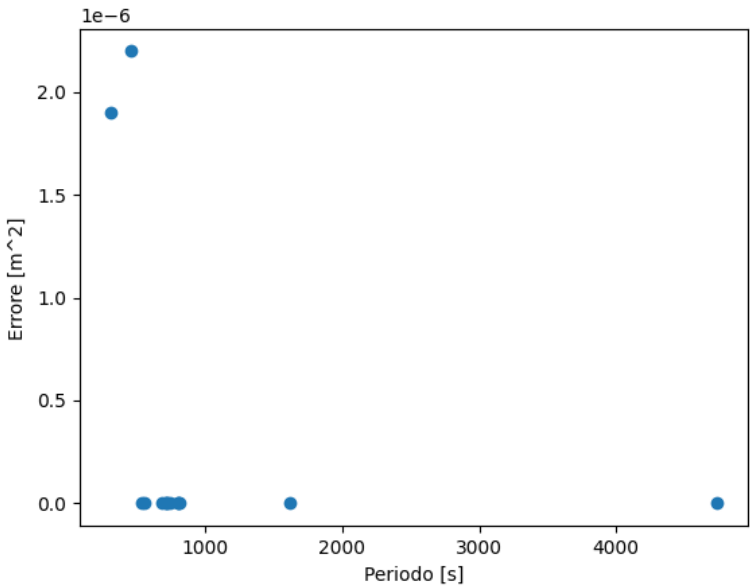


Figura 41: Regione paretiana per problema da 10 punti di tabella 23

- $e(t) = \sin(0.0125t)$ sinusoide con periodo pari a: $T = 502.4s$

Tabella 24: periodo da individuare uguale a 502.4s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	602.0	2.7	5.1e−7
20	801.9	6.0	0.0
30	743.0	2.2	0.0

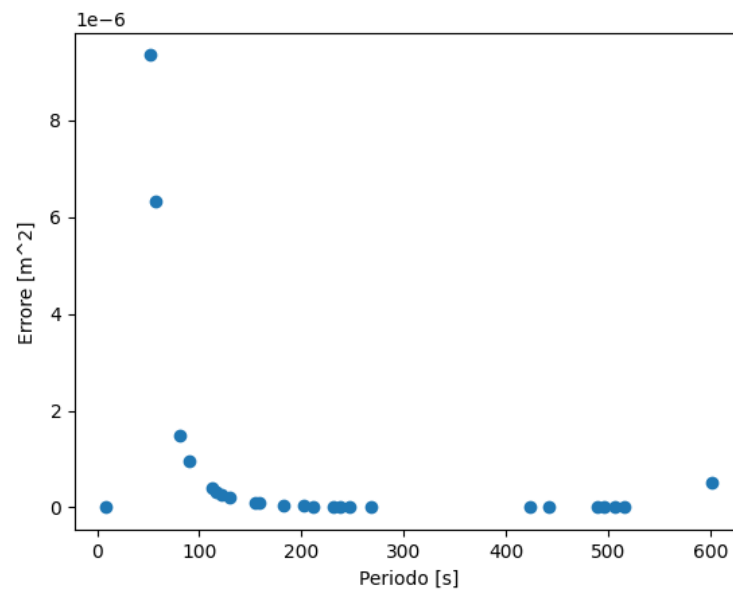


Figura 42: Regione paretiana per problema da 10 punti di tabella 24

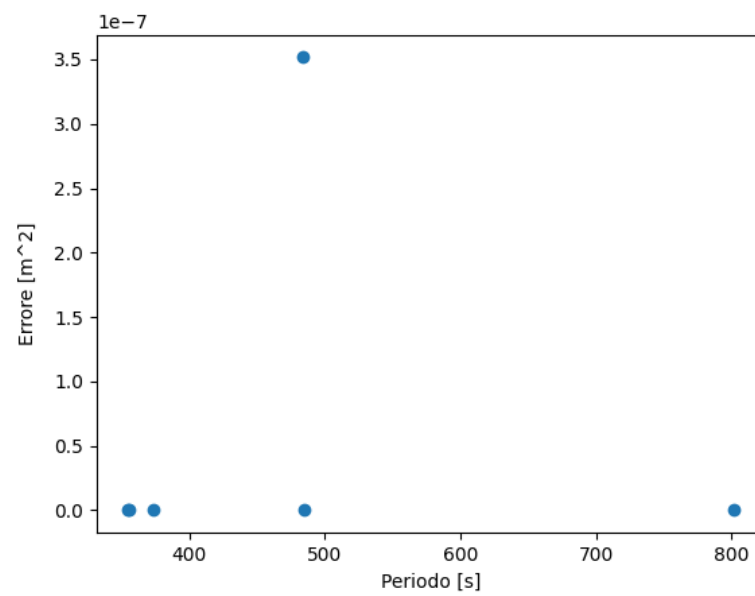


Figura 43: Regione paretiana per problema da 20 punti di tabella 24

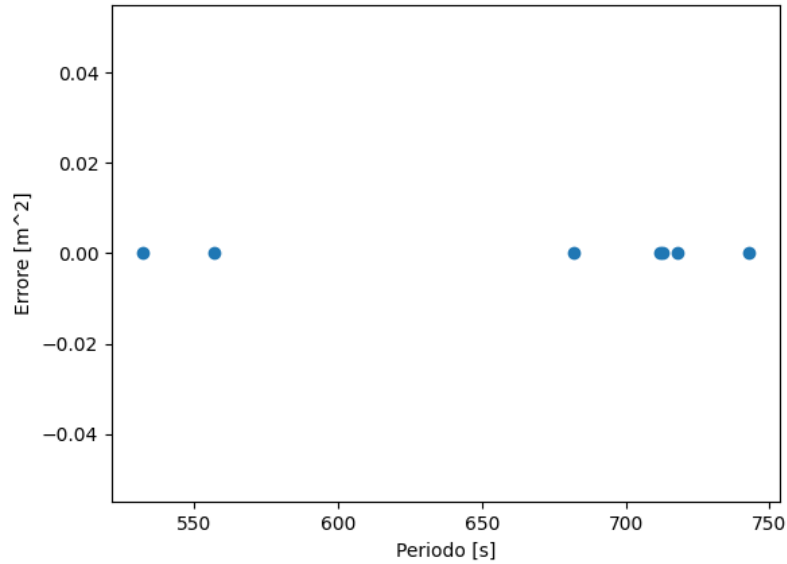


Figura 44: Regione paretiana per problema da 30 punti di tabella 24

- $e(t) = \sin(0.8t)$ sinusoide con periodo pari a: $T = 7.85s$

Tabella 25: periodo da individuare uguale a 7.85s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	7.8	1.8	5.0e−9
20	8.0	4.1	1.1e−2
30	8.0	2.5	2.5e−2

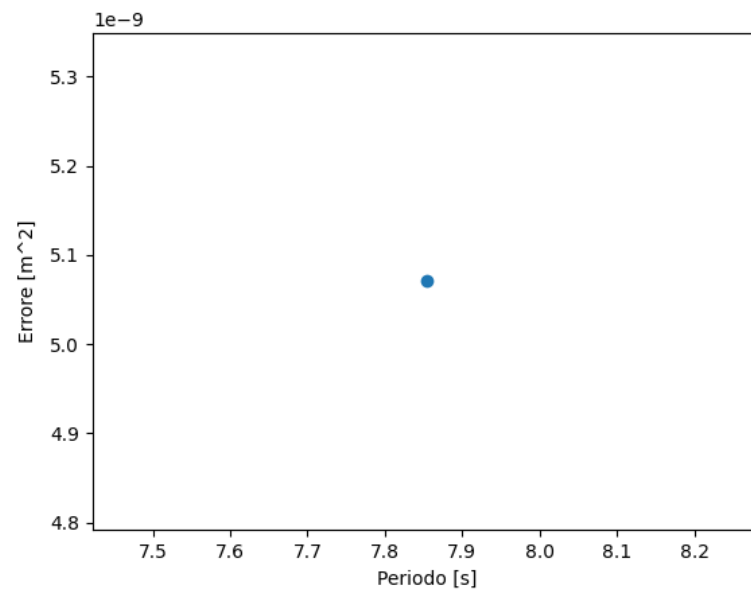


Figura 45: Regione paretiana per problema da 10 punti di tabella 25

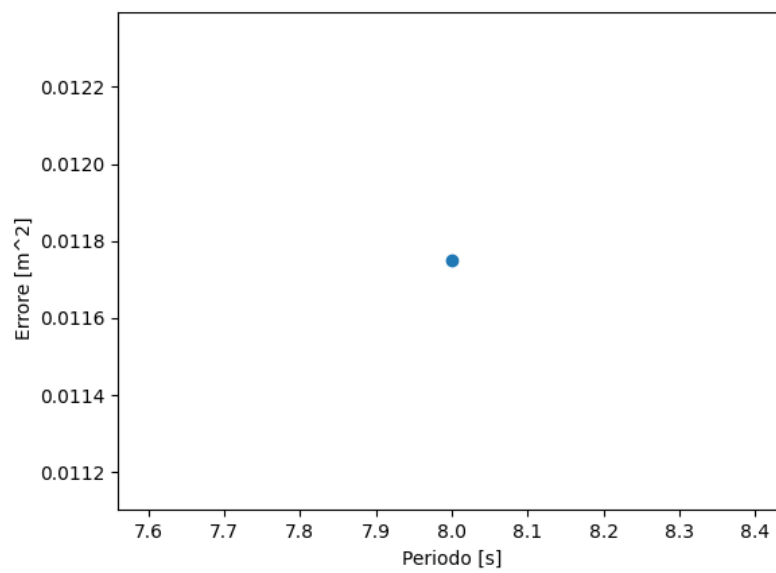


Figura 46: Regione paretiana per problema da 10 punti di tabella 25

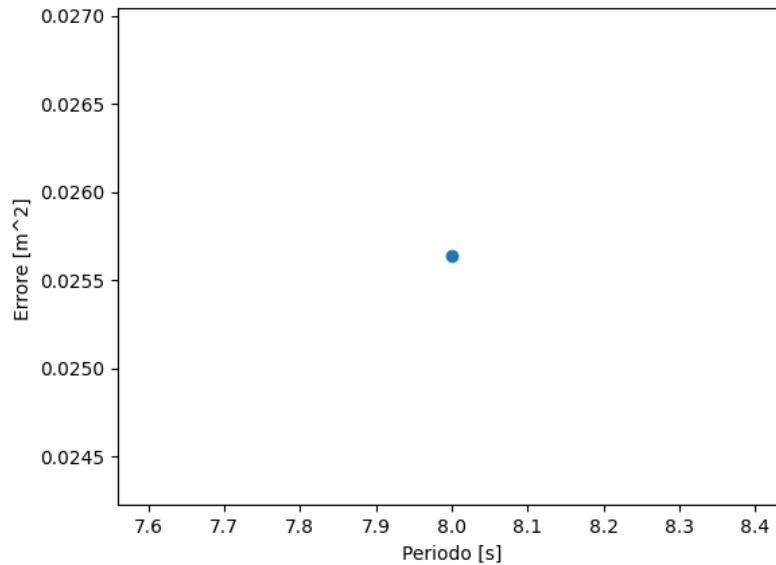


Figura 47: Regione paretiana per problema da 30 punti di tabella 25

L'esecuzione dell'algoritmo è caratterizzato da uno scostamento medio dal periodo da individuare di $610.0s$ e una mediana di $299.3s$ ed infine un tempo medio di esecuzione di $2.5s$ e una mediana di $2.2s$.

È possibile provare a diminuire il tempo di esecuzione modificando l'algoritmo di controllo ed esplorazione, aggiungendo come condizione di aumento del passo, l'individuazione di una soluzione dal livello maggiore di un livello di soglia. Questo permette una velocizzazione dell'esplorazione saltando tutti quei periodi di partenza lontani dal periodo della sinusoide da individuare.

Inoltre si può notare che l'errore delle soluzioni individuate si distribuisce in un range pari a $[0.0, 2.5e-2]$, questo potrebbe suggerire la definizione di livelli dai range più stringenti come ad esempio:

- Livello 1: errore $< 1e-9$
- Livello 2: errore $< 1e-7$
- Livello 3: errore $< 1e-5$
- Livello 4: errore $< 1e-3$
- Livello 5: errore $< 1e-1$

- Livello 6: errore < 1
- Livello 7: errore < 3
- Livello 8: errore < 5
- Livello 9: errore ≥ 5

Infine un'ulteriore modifica per l'accelerazione dell'algoritmo è agire sul parametro k della $f(x)$ in riga 9, algoritmo 1, fissandolo pari a 10 il doppio del valore fissato precedentemente.

Effettuando questa modifica e scegliendo come livello di soglia il quinto, si ottengono i seguenti risultati:

- $e(t) = \sin(0.0010t)$ senoide con periodo pari a: $T = \frac{2\pi}{\omega} = \frac{2\pi}{0.0010} = 6280s$

Tabella 26: periodo da individuare uguale a 6280s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	4610.8	1.2	9.8e-7
20	4461.0	1.2	0.0
30	4906.0	1.2	0.0

- $e(t) = \sin(0.0013t)$ senoide con periodo pari a: $T = 4830.7s$

Tabella 27: periodo da individuare uguale a 4830.7s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	4582.0	1.0	4.3e-8
20	4942.0	1.2	0.0
30	4461.0	1.2	0.0

- $e(t) = \sin(0.005t)$ senoide con periodo pari a: $T = 1256s$

Tabella 28: periodo da individuare uguale a 1256s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	1683.0	1.3	8.2e-6
20	1224.0	1.8	0.0
30	4320.4	2.3	0.0

- $e(t) = \sin(0.0125t)$ senoide con periodo pari a: $T = 502.4s$

Tabella 29: periodo da individuare uguale a 502.4s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	498.1	1.8	3.1e-7
20	706.0	2.2	0.0
30	756.0	1.4	0.0

- $e(t) = \sin(0.8t)$ senoide con periodo pari a: $T = 7.85s$

Tabella 30: periodo da individuare uguale a 7.85s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	7.8	1.1	5.7e-5
20	8.0	1.2	1.1e-2
30	8.0	1.2	2.5e-2

Così ottenendo uno scostamento medio di 648.1s e con mediana pari a 248.7s e un tempo medio di esecuzione di 1.4s con mediana pari a 1.2s. Si ottiene così una diminuzione dei tempi di esecuzione.

Capitolo 4

Sviluppo dell'algoritmo

Date n osservazioni composti da k valori, dove k è il numero di satelliti, l'algoritmo di assegnamento ha il compito di definire e valutare gli assegnamenti di n valori, presi da n osservazioni differenti. Lo scopo è di individuare gli assegnamenti che identificano le sinusoidi corrispondenti al moto dei satelliti.

Tabella 31: Esempio di struttura dati per le osservazioni

	1° satellite	2° satellite	...	k-esimo satellite
1° osservazione	val	val	...	val
2° osservazione	val	val	...	val
...	val	val	...	val
$n -esima$ osservazione	val	val	...	val

4.1 Struttura e approccio

Il concetto alla base è quello di generare tutte le permutazioni possibili delle n osservazioni, e di valutarle tramite i parametri restituiti dal modulo di interpolazione. Il numero delle permutazioni è pari a $(k!)^n$.

La permutazione di una singola osservazione produce $k!$ elementi, ognuno di questi elementi deve essere assegnato a tutti i $k!$ elementi delle successive osservazioni fino alla n -esima osservazione:

$$\prod_{i=1}^n k! \quad (10)$$

Il numero delle permutazioni evidenzia il costo estensivo del processo e comporta l'applicazione di metodologie atti a ridurre il numero di permutazioni valutate per l'individuazione degli assegnamenti corretti in tempo utile.

Per queste motivazioni, ho implementato un algoritmo divide et impera, che divide il problema di assegnamento in sottoproblemi dal numero di osservazioni minore e unisce i risultati ottenuti nei vari nodi fino ad ottenere la soluzione del problema originario.

4.1.1 Creazione dei sottoproblemi

Per la generazione dei sottoproblemi, sfrutto la tecnica della ricorsione, generando un albero di chiamate dove ogni nodo corrisponde ad un sottoproblema. I nodi figli sono definiti in un nodo, suddividendo a metà il numero di osservazioni dati in input, si effettua questo processo finché il numero di osservazioni in un nodo è minore o uguale ad un numero di osservazioni fissato, determinando le foglie.

Una volta che le foglie individuano una assegnazione dei punti ritenuta corretta, questa viene restituita al nodo padre che unirà la soluzione con la soluzione individuata dall'altro nodo figlio.

Algoritmo 7: Generatore Albero

```

input : valori Matrice contenente n osservazioni da k valori
output: Struttura dati contenente n osservazioni da k valori
1 nOss ← NumeroOsservazioni(valori)
2 if nOss ≤ dimensioneFoglia then
3   | return OttimizzaFoglia(valori)
4 end
5 return OttimizzaNodo(GeneraAlbero(valori[0 : nOss/2]),
   GeneraAlbero(valori[nOss/2 : nOss]))

```

4.1.2 Gestione dei nodi

Il compito di un nodo è quello di unire gli assegnamenti individuati nei nodi figli. Questo viene effettuato tramite la permutazione delle colonne di uno delle due matrici, unendo per le colonne, la matrice non permutata con la matrice permutata.

Le colonne della matrice risultante sono valutate ad una ad una tramite il modulo di interpolazione, non appena si individua una colonna dall'errore con livello inferiore ad un livello soglia (riferimento ai livelli dell'algoritmo di selezione di una soluzione in 3.2.6), la si memorizza nella matrice da restituire, togliendo le colonne

che compongono la colonna memorizzata dalla matrice da permutare e dalla matrice non permutata.

Nel caso non si trovi una colonna che rispetti il criterio di selezione, si seleziona la colonna con errore dal livello minore individuata nel corso di tutte le permutazioni (da riga 25 a 32 dell'algoritmo 8).

Si effettuano queste operazioni finché non rimane una sola colonna nella matrice da permutare e nella matrice non permutata.

Il caso migliore è identificato nella individuazione della colonna ottimale nella prima iterazione delle permutazioni, dall'inizio fino ad ogni rimozione di colonna, dove l'algoritmo analizza $k - 1$ permutazioni, k rappresenta il numero di colonne. Il caso peggiore è identificato nell'individuazione della colonna ottimale nell'ultima iterazione delle permutazioni dall'inizio fino ad ogni rimozione di colonna, dove l'algoritmo analizza $\sum_{i=0}^k (k - i)!$ permutazioni. Nel caso si valutassero le colonne di una permutazione complessivamente, quindi senza rimozioni di colonne, sarebbe necessario valutare tutte le permutazioni, l'algoritmo analizzerebbe $k!$ permutazioni.

Ho scelto di applicare l'operazione di rimozione per i vantaggi portati dal suo caso migliore e dal fatto che il caso peggiore non abbia un aumento così drastico delle permutazioni da analizzare.

4.1.3 Gestione delle foglie

Le foglie sono caratterizzati da un numero di osservazioni minore o uguale ad un determinato valore (riga 2, algoritmo 7). Questo valore determina il numero di foglie che verranno generati, per esempio con foglie di dimensioni pari a 10 osservazioni su un totale di 20, si hanno $\frac{20}{10} = 2$ foglie, oppure con foglie di dimensione pari a 5, si hanno $\frac{20}{5} = 4$ foglie. Per ridurre il numero di permutazioni valutate, l'obiettivo è quello di generare quante più foglie possibili.

Nella determinazione del numero di osservazioni massimo per una foglia è necessario tenere a mente anche il modulo di interpolazione: non è fattibile o è di difficile esecuzione l'interpolazione di una sinusoide a partire da pochi punti, come un solo punto o due.

Per la specificazione della dimensione dei nodi foglia, ho definito l'algoritmo 9, che fissa il parametro al maggior numero primo che divide il numero di osservazioni totale, scegliendo solo numeri primi maggiori di 2. Nel caso il numero di osservazioni sia un numero primo si valuta il numero precedente ad esso.

Il processo eseguito è quello descritto nella sezione 4.1, con la differenza che il numero di osservazioni in una foglia è minore uguale al parametro fissato per la dimensione della foglia. Si generano le permutazioni di ogni punto per ogni osservazione, effettuando le stesse operazioni di rimozione dei nodi. Non appena si individua una colonna dall'errore con livello inferiore ad un livello soglia (riferimento ai livelli

Algoritmo 8: Gestione nodo

```

input : val1 Matrice contenente  $j$  osservazioni da  $k$  valori
        val2 Matrice contenente  $j$  osservazioni da  $k$  valori
output: Struttura dati contenente  $2j$  osservazioni da  $k$  valori
1 permutazioni  $\leftarrow$  GeneraPermutazioni(val2)
2 colonnaMigliore  $\leftarrow$  [ ]
3 while True do
4   for perm in permutazioni do
5     unione  $\leftarrow$  Unisci(val1, perm)
6     for col in unione do
7       senoide  $\leftarrow$  IndividuaSenoide(col)
8       if LivelloDi(senoide)  $\leq$  livelloSoglia then
9         rimuovi le colonne di val1 e val2 che compongono col
10        rimuovi col da unione
11        permutazioni  $\leftarrow$  GeneraPermutazioni(val2)
12        aggiungi col ad assegnamentoOttimale
13        if NumeroColonne(unione) = 1 then
14          rimuovi le colonne di val1 e val2 che compongono la colonna
15          rimasta in unione
16          aggiungi la colonna rimasta in unione ad
17          assegnamentoOttimale
18          return assegnamentoOttimale
19        end
20        break
21      end
22      if LivelloDi(sin)  $\leq$  LivelloDi(colonnaMigliore) then
23        | colonnaMigliore  $\leftarrow$  col
24      end
25    end
26    rimuovi le colonne di val1 e val2 che compongono colonnaMigliore
27    permutazioni  $\leftarrow$  GeneraPermutazioni(val2)
28    aggiungi col ad assegnamentoOttimale
29    if NumeroColonne(val2) = 1 then
30      | rimuovi le colonne di val1 e val2 rimaste
31      | aggiungi colonnaMigliore ad assegnamentoOttimale
32      | return assegnamentoOttimale
33    end
34  end

```

dell'algoritmo di selezione di una soluzione in 3.2.6), la si memorizza nella matrice da restituire, togliendo la colonna dalla matrice da permutare, queste operazioni vengono effettuate finché non rimane una sigola colonna nella matrice.

Il caso migliore corrisponde all'individuare una colonna ottimale alle prime permutazioni fino ad ogni rimozione, quindi il numero di permutazioni valutate corrisponde a $k - 1$, mentre il caso peggiore corrisponde all'individuazione della colonna ottimale nell'ultima iterazione delle permutazioni dall'inizio fino ad ogni rimozione di colonna, dove l'algoritmo analizza $\sum_{i=0}^{k-1} ((k - i)!)^n$ permutazioni. Nel caso non si effettuino operazioni di rimozione, il numero totale di permutazioni valutate corrisponde a $(k!)^n$. Come nella gestione dei nodi non foglia 4.1.2, per i vantaggi portati dal caso migliore, e un aumento non significativo delle permutazioni nel caso peggiore, ho scelto di applicare le operazioni di rimozione.

Algoritmo 9: Definizione della dimensione dei nodi foglia

```

input  : nOsservazioni numero di osservazioni
output: dimFoglia numero di osservazioni massimo nei nodi foglia
1 dimFoglia  $\leftarrow$  0
2 if nOsservazioni è primo then
3   | nOsservazioni  $\leftarrow$  nOsservazioni - 1
4 end
5 for  $i$  in [3, nOsservazioni] do
6   | if  $i$  è primo ed è un divisore di nOsservazioni then
7     | dimFoglia  $\leftarrow$   $i$ 
8   | end
9 end
10 return dimFoglia

```

Algoritmo 10: Gestione nodo foglia

input : val Matrice contenente j osservazioni da k valori
output: assegnamentoOttimale Struttura dati contenente j osservazioni da k valori

```

1 permutazioni  $\leftarrow$  GeneraPermutazioni(val)
2 while True do
3   for perm in permutazioni do
4     for col in perm do
5       senoide  $\leftarrow$  IndividuaSenoide(col)
6       if LivelloDi(senoide)  $\leq$  livelloSoglia then
7         rimuovi col da perm
8         permutazioni  $\leftarrow$  GeneraPermutazioni(perm)
9         aggiungi col ad assegnamentoOttimale
10        if NumeroColonne(perm) = 1 then
11          aggiungi la colonna rimasta in perm ad
            assegnamentoOttimale
12          return assegnamentoOttimale
13        end
14      break
15    end
16  end
17 end
18 end

```

4.1.4 Valutazione delle prestazioni

La valutazione delle prestazioni del modulo di assegnamento si concentra sull'analisi ed esame dei tempi di calcolo complessivi, tempo medio di calcolo per nodo, tempo medio di calcolo per foglia e percentuale di punti correttamente assegnati.

I punti in input sono riordinati in modo randomico utilizzando il modulo random di python (ref) perciò definito un caso di test, è necessario eseguire per un numero fissato di volte il modulo di assegnamento. In modo da recuperare ed analizzare il comportamento del modulo indipendentemente dalle caratteristiche dell'assegnamento iniziale in ingresso. Il numero di iterazioni fissato è pari a 10.

I casi di test utilizzati si differenziano per numero di osservazioni, numero di satelliti, e caratteristiche delle sinusoidi utilizzate per generare i punti. Il numero di osservazioni permette di verificare il comportamento dell'algoritmo di definizione del numero di osservazioni massimo nelle foglie, algoritmo 9 e differenza in prestazioni dei nodi rispetto alle foglie e viceversa. Il numero di satelliti insieme alle sinusoidi

Tabella 32: caso di test 10 osservazioni

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	260	2.3	128.5	100
2	277	1.7	138	100
3	81	2.6	39	100
4	54	5.1	24	100
5	20	1.5	8	100
6	48	1.1	23	100
7	306	1.7	152	100
8	50	0.8	24	100
9	55	6.8	24	100
10	22	1.1	10	100
10	22	1.1	10	100
media	117.3	2.4	54.95	100

utilizzate permettono di valutare il comportamento in maniera generale del modulo di assegnamento. I tre casi di test di tabelle 32, 33 e 34, sono definiti da tre sinusodi, che coprono il range di periodi definito in 3.2.2:

- $e(t) = \sin(0.314t)$
- $e(t) = 10\sin(0.0027t + \pi)$
- $e(t) = 4\sin(0.00139t + \frac{\pi}{2})$

I tre casi di test si differenziano in numero di osservazioni, che corrispondono rispettivamente a 10, 20, 30.

Gli ultimi due casi di test di tabelle 35 e 36, sono definiti da sinusodi con parametri stabiliti in maniera randomica utilizzando il modulo random di python (ref) ad ogni esecuzione. L'ampiezza viene definito da un insieme $[1, 10]$ mentre il periodo dall'insieme $[1, 5000]$.

Questo permette di sperimentare e verificare il comportamento del modulo di assegnamento in ogni caso possibile e non basare considerazioni ed analisi su sinusodi specifiche, che potrebbero condizionare la bontà dei risultati. Gli ultimi due casi di test si differenziano per numero di satelliti, rispettivamente 3 e 4, mentre il numero di osservazioni è pari a 30.

Il numero di permutazioni da valutare nelle foglie può essere molto maggiore rispetto al numero di permutazioni valutate nei nodi.

Tabella 33: caso di test 20 osservazioni

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	220	3	43.4	90
2	449	2.2	110.7	93.3
3	108	4.5	47.3	90
4	182	3.9	42.4	86.6
5	348	3.2	84.3	86.6
6	149	3.7	34.4	86.6
7	24	3.5	3.2	90
8	384	2.5	94.2	90
9	374	3.5	90.9	90
10	83	1.8	38.6	100
media	232.1	3.1	58.95	90.3

Tabella 34: caso di test 30 osservazioni

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	165	5.9	15.4	66.6
2	90	3.0	8.5	89.9
3	200	8.5	17.5	79.9
4	152	4.0	15.4	74.4
5	108	3.5	10.4	68.8
6	158	7.1	13.4	58.8
7	121	5.3	10.5	68.8
8	96.4	3.7	8.7	58.8
9	103	3.1	10.1	68.8
10	142	5.9	12.6	56.6
media	133	5.0	12.2	69.1

Tabella 35: caso di test 3 satelliti

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	107	4.2	9.6	76.6
2	86	5.0	6.3	95.5
3	125	1.1	14.7	75.5
4	227	18	12	44.4
5	73	4.7	5	73.3
6	103	1.8	11.2	87.7
7	205	3.6	22.4	54.4
8	259	15.7	18.7	54.4
9	32	1.2	2.9	58.8
10	210	6.2	20	64.4
media	142	6.15	12.28	68.5

Tabella 36: caso di test 4 satelliti

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	1207	36	119	69.9
2	638	20	61	73.2
3	269	14	20	88.3
4	297	19.7	19.8	57.5
5	487	7.9	53.9	52.4
6	522	31	37	67.4
7	236	12	18	57.4
8	295	4.8	32	66.6
9	596	22.7	54.6	75
10	1648	28	181	65
media	619	19.61	59.63	67.2

I risultati delle sperimentazioni evidenziano ciò, la maggior parte del tempo di calcolo è impiegato nell'elaborazione delle foglie. Il tempo medio di elaborazione dei nodi aumenta solo all'aumentare dei satelliti rilevati. Si può notare questo nei risultati in tabella 35 e 36, dove passa da 6.15s a 19.61s.

Il tempo di esecuzione complessivo è fortemente influenzato dal numero di foglie definite e dal numero di osservazioni analizzate in esse. Ad esempio nel caso di test di tabella 32 e 33, la dimensione delle foglie può essere minore o uguale a 5 osservazioni e in quei casi di test corrisponde esattamente al valore fissato, poiché il quoziente della divisione tra numero di osservazioni totali e la dimensione massima delle foglie è un multiplo di 2, figura 48. Nel caso di test di tabella 32 si generano due foglie mentre in quello di tabella 33 si generano quattro foglie, perciò a parità di dimensione di foglia, al raddoppiare del numero di foglie, il tempo medio raddoppia.

La differenza tra il caso di test dei risultati di tabella 34 dai casi di test di tabella 32 e 33 è che seppur la dimensione massima delle foglie sia uguale, le foglie definite nel caso di test da 30 osservazioni sono pari a quattro o tre osservazioni, figura 49. Questo va a diminuire il numero di permutazioni valutate all'interno delle foglie, caratterizzando una diminuzione del tempo medio di esecuzione, ma comporta anche una diminuzione della correttezza dell'assegnamento.

Il caso di test dei risultati di tabella 35 è caratterizzato da una generazione randomica delle sinusoidi per la definizione dei punti da assegnare, nonostante questo i tempi medi e la correttezza dell'assegnamento non si scostano di molto dai valori medi di tabella 34 con stesso numero di osservazioni e sinusoidi fissate.

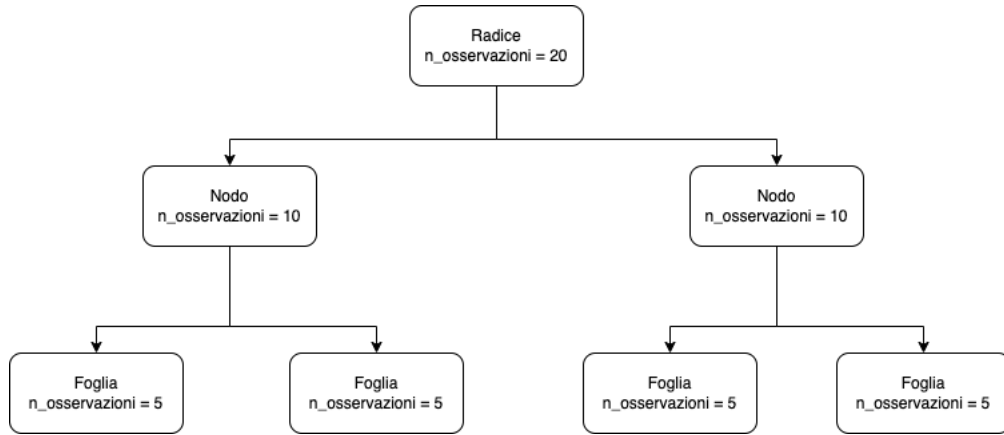


Figura 48: Albero dei sottoproblemi per il caso di test 33

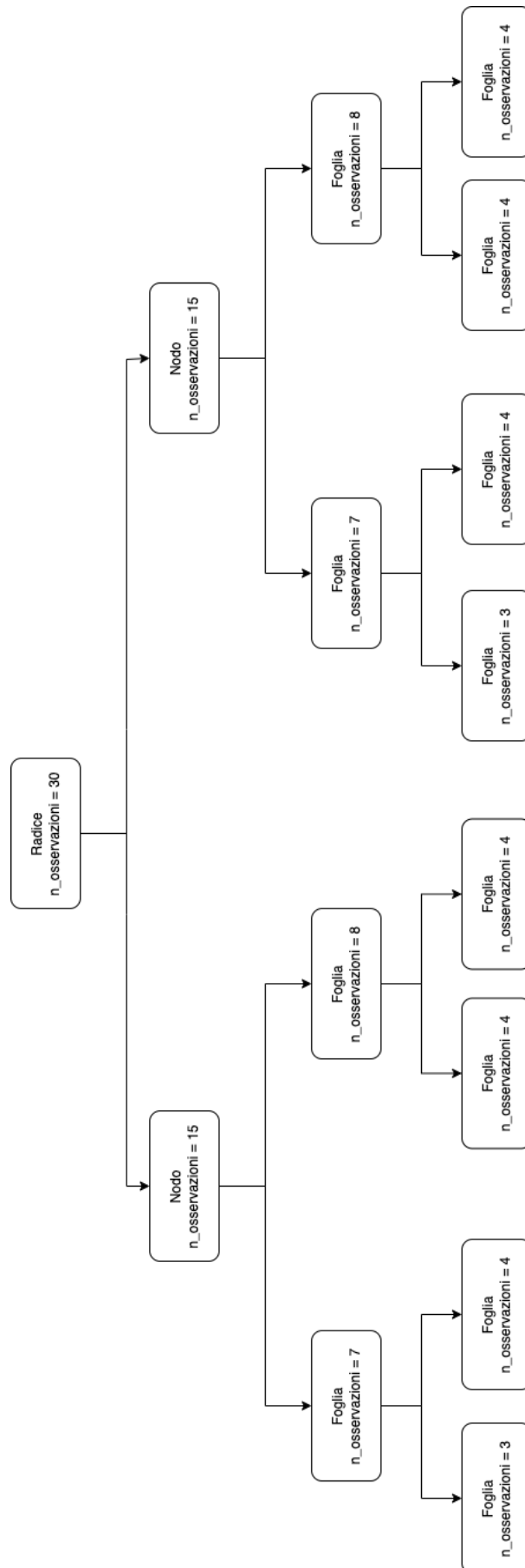


Figura 49: Albero dei sottoproblemi per il caso di test 34

4.1.5 Miglioramenti

Il threading è un miglioramento implementabile. Un thread o thread di esecuzione, è una suddivisione di un processo in due o più istanze o sottoprocessi che vengono eseguiti concorrentemente. L'esecuzione parallela nel caso del modulo di assegnamento non comporta alcuna problematica, siccome l'elaborazione dei nodi e delle foglie sono indipendenti l'uno dall'altra. È possibile quindi tramite l'utilizzo dei thread, ridurre drasticamente il tempo di esecuzione, tramite l'esecuzione parallela di interi rami o singoli nodi o foglie.

Bibliografia

- [1] Steven G. Johnson, The NLOpt nonlinear-optimization package
- [2] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*, eds. S. Gomez and J.-P. Hennart (Kluwer Academic: Dordrecht, 1994), da pagina 51 a 67.
- [3] Dr. David R. Williams, Planetary Fact Sheets, NASA Goddard Space Flight Center 2016
- [4] Sartoretti P. and Schneide, J. "On the detection of satellites of extrasolar planets with the method of transits" 1999, da pagina 553 a 560, SAO/NASA Astrophysics Data System.