Università degli Studi di Napoli "Federico II"

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

ELABORATO D'ESME PER MODELLI E METODI DELLA RICERCA OPERATIVA

GRUPPO 17- TRACCIA 15 - OPTIMAL LOCATION OF SECURITY DEVICES
Weighted Demand Covering Problem (WDCP)

Prof. Ing. Claudio Sterle

Studenti:
Alberto Guglielmo Matr. N38000261
Gianmarco Iodice Matr. N38000260

ANNO ACCADEMICO 2023 / 2024

Indice

Introduzione **3**

CAPITOLO 1. DESCRIZIONE DEL MODELLO MATEMATICO (WDCP) **5**

1.1 Modello 5

CAPITOLO 2. CODICE PYTHON-GUROBI **7**

2.1 Input dei dati 7

2.2 Modello di ottimizzazione 8

2.3 Analisi dei Risultati di Ottimizzazione 8

CAPITOLO 3. TEST DEL MODELLO E RISULTATI **9**

3.1 Esempio 1 10

3.2 Esempio 2 12

3.3 Esempio 3 14

CAPITOLO 4. DESCRIZONE E CALCOLO DI UB E LB **17**

4.1 Upper Bound 17

4.2 Lower Bound 19

4.3 Test calcolando gli UB e LB 19

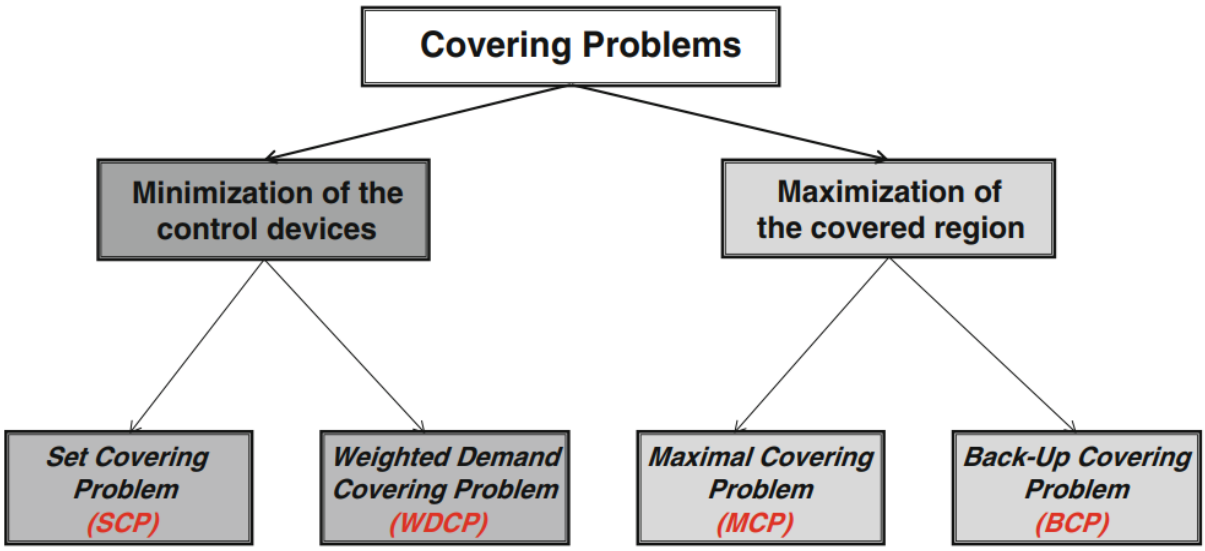
4.4 Valutazione dell'UB e del LB considerati 20

Introduzione

In questo elaborato viene affrontato il problema dell'ottimale posizionamento dei sensori per la sicurezza delle infrastrutture ferroviarie. Viene analizzato il progetto europeo METRIP (MEthodological Tool for Railway Infrastructure Protection), che si propone di sviluppare uno strumento metodologico per aumentare la protezione delle risorse infrastrutturali sensibili.
Il testo evidenzia che un elemento chiave del METRIP è il Modulo di Ottimizzazione (OM), il cui principale compito è determinare la posizione ottimale dei dispositivi di protezione all'interno dell'asset ferroviario. Questo problema è descritto come il "problema del posizionamento del sensore"; di particolare importanza è l'utilizzo della programmazione lineare intera (ILP) per risolvere i modelli implementati.
L'Analisi della copertura di un dispositivo può essere schematizzata attraverso l'area di copertura, cioè la porzione di territorio che può essere controllata. È definito da due parametri:
1. angolo di copertura, espresso in gradi (0°÷360°), all'interno del quale il dispositivo è attivo;
2. raggio di copertura r, distanza massima alla quale il dispositivo è ancora efficace.
Sul set da monitorare possiamo individuare o un singolo dispositivo con un determinato orientamento, oppure più dispositivi con orientamenti diversi.
L'Analisi della copertura consiste nel determinare quali sono i punti di R (punti della mappa analizzata) che possono essere controllati da un dispositivo posizionato in una determinata posizione e avente un certo angolo di copertura. Nel seguito questo insieme di punti verrà chiamato S, $S \subseteq R$.
I "problemi di localizzazione" richiedono di determinare la migliore posizione per dispositivi di sicurezza in base a criteri prestazionali e vincoli operativi.
Il Modulo di Ottimizzazione di METRIP esegue attività e l'analisi della copertura e la selezione del modello di copertura.
Il problema di copertura può essere classificato in base a due obiettivi principali:

- minimizzare il numero o il costo totale dei dispositivi di controllo da localizzare
- massimizzare la regione coperta da tali dispositivi.

Il primo ordine di problemi di copertura vuole determinare il numero di dispositivi di controllo da localizzare, minimizzando il costo totale di installazione e coprendo tutti i punti della regione di interesse o un sottoinsieme di essi. All'interno della prima classe è possibile operare la classificazione dei punti della regione in due gruppi, punti importanti e generali.
Un punto può essere classificato come importante se particolarmente vulnerabile o un punto per cui un malintenzionato dovrà per forza transitare. I punti importanti, al contrario di punti generali, devono essere obbligatoriamente controllati. Se tutti i punti hanno la stessa importanza, il problema viene definito problema di copertura dell'insieme (SCP), altrimenti si parla di Problema di Copertura della Domanda Ponderata (WDCP).


Covering Problems
Minimization of the control devices
Maximization of the covered region
Set Covering Problem (SCP)
Weighted Demand Covering Problem (WDCP)
Maximal Covering Problem (MCP)
Back-Up Covering Problem (BCP)

CAPITOLO 1. DESCRIZIONE DEL MODELLO MATEMATICO (WDCP)

1.1 Modello

Per descrivere il modello matematico del problema WDCP occorre definire:
- $R = \{1, ..., |R|\}$ insieme di punti che rappresentano la regione di interesse.

- $L' = \{1, ..., |L'|\}$ insieme di posizioni potenziali del dispositivo con orientamenti;

- s_j valore del flag definito per ogni j, $j \in R$. È uguale a 1 se il punto j deve essere controllato obbligatoriamente, 0 altrimenti;

- h_i costo di installazione in un potenziale punto i, $i \in L'$. (consideriamo tutti i costi di installazione uguali e pari a 1)

- α parametro compreso tra 0 e 1 che regola il posizionamento di un nuovo dispositivo nel caso in cui un numero "significativo" di punti generali della regione di interesse sia controllato.

Inoltre, verranno utilizzate le seguenti variabili:

- $y_i = \{0, 1\}$: variabile binaria associata a ogni potenziale posizione del dispositivo i, $i \in L'$, con un orientamento specifico. È uguale a 1 se nella posizione i è installato un dispositivo, 0 altrimenti.

- $x_j = \{0, 1\}$: variabile binaria associata a ogni punto j da coprire, $j \in R$. è uguale a 1 se il punto j è coperto, 0 altrimenti.

È importante sottolineare che, dato che un dispositivo può essere posizionato in una potenziale posizione utilizzando diversi orientamenti, ogni variabile y_i , $i \in L'$, corrisponde a un dispositivo localizzato in un determinato punto avente un determinato orientamento.

Weighted Demand Covering Problem (*WDCP*)
The ILP model for the weighted demand covering problem is:

$$Min\, z = \sum_{i \in L'} y_i - \alpha \sum_{j \in R} (1 - s_j) x_j \quad (4)$$

s.t.

$$\sum_{i \in L'} c_{ij}\, y_i \geq 1 \quad \forall j \in R | s_j = 1 \quad (5)$$

$$\sum_{i \in L'} c_{ij}\, y_i \geq x_j \quad \forall j \in R | s_j = 0 \quad (6)$$

$$y_i = \{0, 1\} \quad \forall i \in L' \quad (7.a)$$

$$x_j = \{0, 1\} \quad \forall j \in R \quad (7.b)$$

La funzione obiettivo (4) è composta da due termini. Il primo termine minimizza il numero di dispositivi di controllo da posizionare per coprire tutti i punti importanti della regione. Il secondo termine cerca di localizzare un dispositivo di controllo aggiuntivo se la sua installazione aumenta il numero di punti generali controllati di un valore minimo di soglia definito dal parametro α. Per definire il valore α, si installa un nuovo dispositivo di controllo solo se vengono coperti N* punti aggiuntivi; in tal caso il valore di α deve soddisfare le due seguenti condizioni:

$$1-\alpha \text{ x } N^* < 0$$
$$1-\alpha \text{ x } (N^*-1)>0$$

Si noti che se α = 0 non verrà aggiunta nessuna telecamera non indispensabile.
Il vincolo (5) impone che ogni punto importante j, j ∈ R, deve essere coperto almeno da un dispositivo i, i ∈ L'. Il vincolo (6) impone che un punto generico sia coperto solo nel caso in cui sia stato installato un dispositivo in grado di controllarlo. I vincoli (7a) e (7b) sono vincoli binari per y_i , i ∈ L' e x_j ,j ∈ R'.

L'analisi di copertura permette di generare la Matrice di Copertura (C), una matrice binaria in cui le righe corrispondono agli elementi dell'insieme L', cioè a tutte le telecamere potenziali con rispettivo orientamento, e le colonne corrispondono ai punti dell'insieme R da coprire. Le sue dimensioni sono quindi (n × |L|,|R|), dove n = 360/(angolo di copertura della telecamera).

Il suo elemento generico c_{ij} è uguale a 1 se il dispositivo i, i ∈ L', può coprire il punto j, j ∈ R, (cioè, in altre parole, se il punto j appartiene all'insieme S′ del dispositivo i), altrimenti è uguale a 0. La matrice di copertura è l'input fondamentale per tutti i modelli ILP di copertura utilizzati in METRIP.

Ai fini della risoluzione del problema è stato introdotto un ulteriore vincolo che permette di assegnare ai punti visualizzati dalle telecamere il flag 1.

$$c_{ij} * y_i \leq x_j \;\; \forall\, i \;\in L' \; e \; \forall\, j \;\in R$$

La funzione obiettivo, infatti, tende ad inserire nella soluzione ottima più punti possibili. Questo non avviene quando i termini α o (1-s[j]) si annullano; in questo caso la x[j] può assumere un qualsiasi valore. Affinché ciò non si verifichi è stato aggiunto questo ulteriore vincolo.

CAPITOLO 2. CODICE PYTHON-GUROBI

2.1 Input dei dati

In [ ]:

```
import gurobipy as gp
from gurobipy import GRB
import numpy as np

#leggo da file la matrice di copertura
mod = gp.Model('WDCP')

file = open('TEST2.txt', 'r')
content = file.readlines()
A = []
for i in content:
    array = []
    parse = i.split(",")
    for j in parse:
        array.append(float(j))
    A.append(array)
file.close()
A = np.array(A)




file2 = open('TEST2sj.txt', 'r')
content = file2.readline()
pars1=content.split(",")
s = []
for i in pars1:
    s.append(float(i))
file2.close()
s = np.array(s)



numero_righe=len(A)
numero_colonne=len(A[0])


print('A: ',A)
print('righe di A: ',numero_righe)
print('colonne di A: ',numero_colonne)
print('s: ',s,len(s))

if (len(s)!=numero_colonne):
    print('errore')
```

La matrice di copertura A e il vettore dei punti importanti s vengono generati in uno script di MATLAB, che sarà successivamente mostrato e dipenderà dal caso analizzato.

2.2 Modello di ottimizzazione

In [4]:
```python
#parametro alpha per i punti generali
alpha=1;

#creazione del modello
mod = gp.Model('WDCP')
L = range(numero_righe)
R = range(numero_colonne)
y = mod.addVars(L,vtype=GRB.BINARY,name="y")
x = mod.addVars(R,vtype=GRB.BINARY,name="x")


# Creazione dell'obiettivo
obj = mod.setObjective(gp.quicksum(y[i] for i in L) - alpha * gp.quicksum((1-s[j])*x[j] for j in R), GRB.MINIMIZE)


#vincoli
for j in R:
    if s[j]==1: #aggiungo per forza i punti importanti
        mod.addConstr(gp.quicksum(A[i,j]*y[i] for i in L) >= 1)
    else:     #altrimenti, il punto è generale e la sua appartenza alla soluzione dipende da alpha
        mod.addConstr(gp.quicksum(A[i,j] * y[i] for i in L) >= x[j])
    for i in L:
        mod.addConstr((A[i,j]*y[i])<=x[j])  #se posiziono la telecamera devo segnare i punti coperti

# Risoluzione del problema
mod.optimize()

# Stampa dei risultati

for i in L:
    print(f"y_{i+1} =", int(y[i].x))

for j in R:
    print(f"x_{j+1} =", int(x[j].x))


```

2.3 Analisi dei Risultati di Ottimizzazione

In [14]:
```python
puntivisti=0;

for j in R:
    puntivisti=puntivisti+int(x[j].x)

puntiimportanti=0;

for j in range(numero_colonne):
    puntiimportanti=puntiimportanti+s[j]

telecameremesse=0;
for i in L:
    telecameremesse=telecameremesse+int(y[i].x)

print("numero di posizioni osservate: ",puntivisti) #punti coperti
print("numero telecamere posizionate: ",telecameremesse)
print("numero punti importanti: ",puntiimportanti)

```

Il codice fornisce una panoramica quantitativa dei risultati ottenuti, contando il numero effettivo di punti osservati, il numero totale di punti importanti e il numero di telecamere posizionate.

CAPITOLO 3. TEST DEL MODELLO E RISULTATI

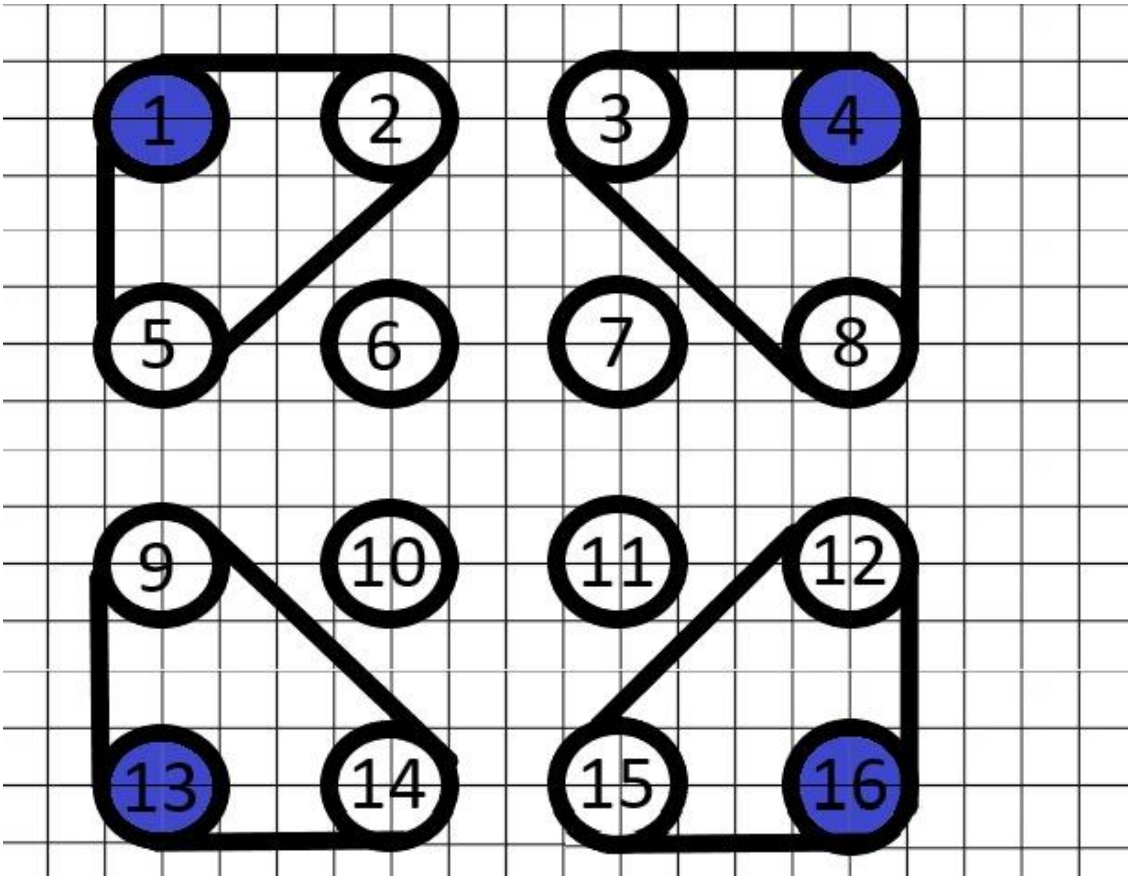
Il modello di copertura è stato sperimentato su tre casi test nei quali i dispositivi da localizzare sono telecamere a prospettiva fissa.

– Il primo scenario rappresenta una stanza di una stazione ferroviaria di dimensione 4x4. In questa configurazione, le telecamere possono essere posizionate solo negli angoli. La caratteristica principale di questo caso è l'assenza di ostacoli nell'area sotto controllo. Inoltre, si è considerato che non sia possibile visualizzare in alcun modo l'area centrale della stanza, poiché tutte le possibili telecamere posizionate negli angoli hanno un raggio di copertura unitario. Questo implica che la zona centrale rimane fuori dalla portata visiva di tutte le telecamere.

– Il secondo scenario rappresenta la planimetria di una stazione ferroviaria di dimensione quadrata 11x11. Analogamente al caso precedente, si è trascurata la presenza di ostacoli nell'area sotto controllo. Tuttavia, in questa situazione, la visione delle telecamere è leggermente superiore a 180°. Su ogni pilastro possono essere installate due telecamere: una telecamera analizza i 180° superiori, mentre l'altra analizza i 180° inferiori. In dettaglio, ciascuna telecamera può visualizzare 3 punti su 4 nella zona circostante (escluso il pilastro), considerando un raggio unitario. La possibile presenza di due telecamere per pilastro rappresenta una strategia per migliorare la copertura al contrario del primo scenario. Le telecamere possono essere installate su nove pilastri, si forniranno risultati dell'ottimizzazione con diversi raggi e variando il parametro α.

– Il terzo scenario raffigura la planimetria di una stazione ferroviaria di dimensioni quadrate pari a 32x32. In contrasto con i casi precedenti, in questa situazione non è stata trascurata la presenza di ostacoli nell'area sotto controllo. Per modellare la presenza degli ostacoli, è stata effettuata una scelta casuale attraverso uno script MATLAB. Nell'implementazione di questo script, sono stati impostati a 0 alcuni punti visualizzati dalle rispettive telecamere, riflettendo così l'influenza degli ostacoli sulla visibilità. La presenza di ostacoli ha introdotto delle zone non visibili dalle telecamere, comportando una limitazione nella copertura rispetto ai casi precedenti in cui la visibilità era completa. Questa simulazione mira a riflettere in modo più realistico le sfide che potrebbero emergere in ambienti reali, dove la presenza di ostacoli può influire significativamente sulla capacità delle telecamere di monitorare l'area circostante. La visione delle telecamere come nel caso precedente è leggermente superiore a 180° e possono essere istallate su 100 pilasti. Si forniranno risultati dell'ottimizzazione al variare del raggio e al variare di α.

3.1 Esempio 1



MATRICE DI COPERTURA
si noti che in questo caso la telecamera può osservare anche il punto in cui essa è posizionata

```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1
```

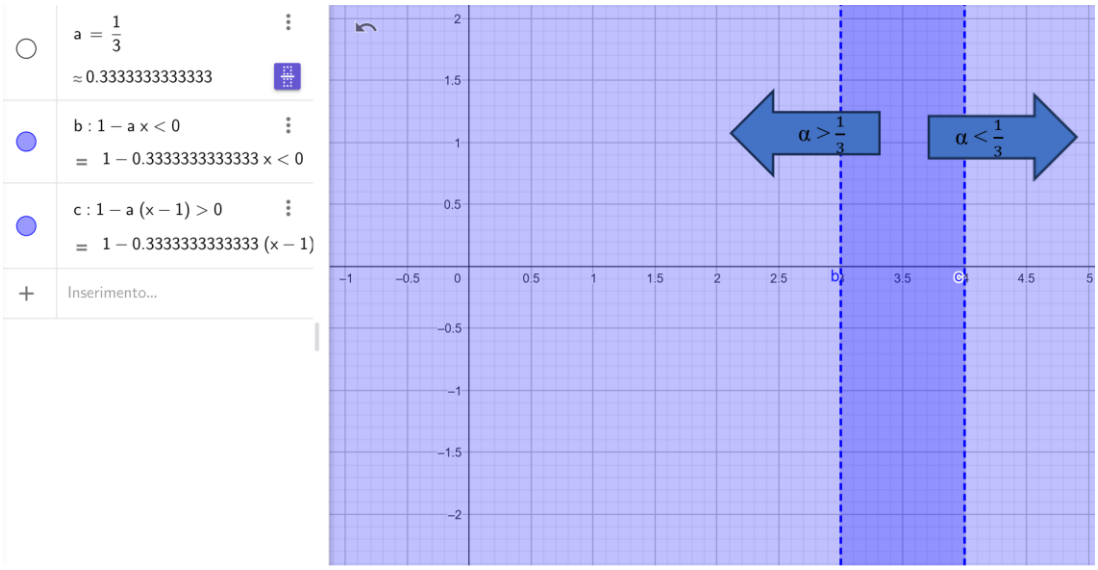
VETTORE DEGLI ELEMENTI IMPORTANTI (posizione 2,15,16)

```
0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1
```

In questo esempio il raggio della telecamera è fissato ad 1 e ci si aspetta sempre $x_6=x_7=x_{10}=x_{11}=0$ poiché' non fisicamente copribili dal momento che il raggio delle telecamere è unitario.
Studiamo ora per quali valori di α le telecamere si accedono.
Se una telecamera copre N* punti aggiuntivi e il valore di α soddisfa le due seguenti condizioni, la telecamera si accenderà.

$$1-\alpha \times N^* < 0$$
$$1-\alpha \times (N^*-1)>0$$

Al variare del parametro α varia l'intersezione con l'asse delle ascisse, su quest'ultimo vengono rappresentati i valori di N*; se l'intersezione tra le disequazioni è compresa tra 1 e 3 allora le telecamere che coprono N* punti aggiuntivi verranno installate. In questo caso le telecamere 4(y_2) e13(y_3) verranno installate per tutti i valori di $\alpha>\frac{1}{3}$.



α=0	α=0.3334	α=1
y_1 = 1 y_2 = 0 y_3 = 0 y_4 = 1 x_1 = 1 x_2 = 1 x_3 = 0 x_4 = 0 x_5 = 1 x_6 = 0 x_7 = 0 x_8 = 0 x_9 = 0 x_10 = 0 x_11 = 0 x_12 = 1 x_13 = 0 x_14 = 0 x_15 = 1 x_16 = 1	y_1 = 1 y_2 = 1 y_3 = 1 y_4 = 1 x_1 = 1 x_2 = 1 x_3 = 1 x_4 = 1 x_5 = 1 x_6 = 0 x_7 = 0 x_8 = 1 x_9 = 1 x_10 = 0 x_11 = 0 x_12 = 1 x_13 = 1 x_14 = 1 x_15 = 1 x_16 = 1	y_1 = 1 y_2 = 1 y_3 = 1 y_4 = 1 x_1 = 1 x_2 = 1 x_3 = 1 x_4 = 1 x_5 = 1 x_6 = 0 x_7 = 0 x_8 = 1 x_9 = 1 x_10 = 0 x_11 = 0 x_12 = 1 x_13 = 1 x_14 = 1 x_15 = 1 x_16 = 1
Z=2	Z=0.994	Z=-5

I tempi di ottimizzazione riportati dal software per tutti i casi sono pari a 0.00sec

Come è evidente, per α=1 vengono installate non solo le telecamere necessarie per coprire i punti obbligatori, ma anche quelle che coprono 1 punto aggiuntivo, ovvero verranno installate anche y2 e y3. Tuttavia, in questo particolare caso con α=0.334, poiché α>1/3, si installeranno le telecamere che coprono 3 punti aggiuntivi. Pertanto, il risultato rimane invariato.

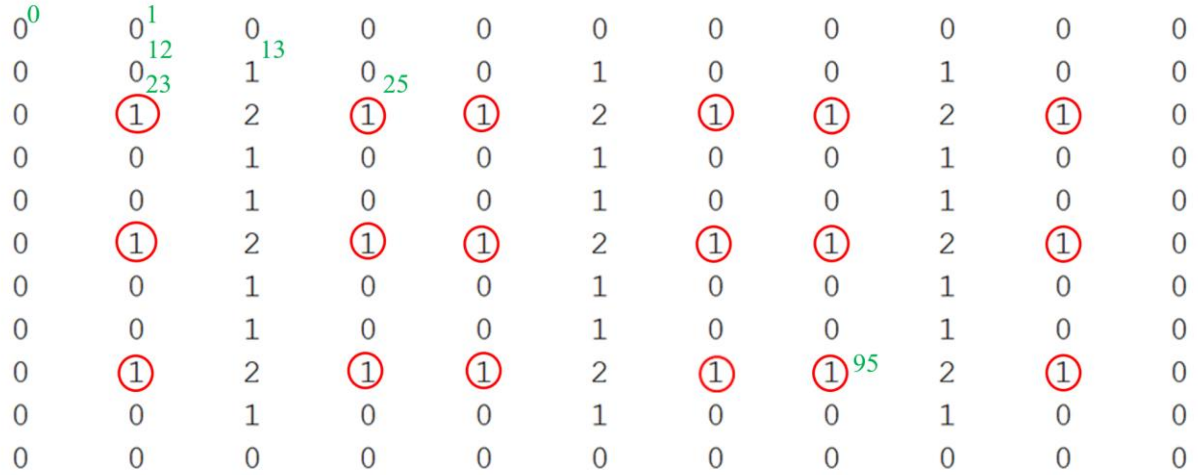
3.2 Esempio 2

Il secondo scenario riguarda una stazione ferroviaria 11x11 senza ostacoli, con telecamere che hanno una visione appena superiore a 180°. Ogni pilastro può ospitare fino a due telecamere, una che copre i 180° superiori e l'altra i 180° inferiori. Ogni telecamera, con un raggio unitario, copre 3 punti su 4 nella zona circostante, escludendo il pilastro. L'ottimizzazione coinvolge la variazione del raggio e del parametro α, cercando di massimizzare l'efficacia della sorveglianza in base alle caratteristiche delle telecamere e alla loro disposizione.

Per il calcolo della matrice di copertura:

```
lato=11;
telecamere_su_lato=((lato-2)/3);
telecamere=telecamere_su_lato^2;
A=zeros(lato,lato);
raggio=1;
C=zeros(telecamere*2,lato*lato);  %matrice di copertura

for i =1:telecamere_su_lato
    for j =1:telecamere_su_lato
     A(i*3,j*3)=2;%assegno telecamere alla mappa
    end
end


for i =1:lato
    for j =1:lato
        for angolo =1:2
           if(A(i,j)==2) %scorro la matrice per cercare le telecamere
              for k =1:lato
                   for w =1:lato  %scorro la matrice per trovare un punto che la mia telecamera copre
                        if((i-k)^2+(j-w)^2<=raggio^2 && (i-k)^2+(j-w)^2~=0)

                            if(angolo==1 && k<=i)                       % guardo sopra
                                A(k,w)=1;                               %segno sulla mappa che ho visto il punto
                                C((i+j/3-3),k*lato+w-lato)=1;  %inserisco l'elemento nella matrice di copertura
                            end
                            if(angolo==2 && k>=i)                       % guardo sotto
                                A(k,w)=1;                               %segno sulla mappa che ho visto il punto
                                C((i+j/3-3)+telecamere,k*lato+w-lato)=1;  %inserisco l'elemento nella matrice di copertura
                            end
                        end
                   end
              end
           end
        end
    end
end

writematrix(C, "TEST2.txt")
```

Dal momento che il codice può risultare di difficile comprensione, si allega nel caso di raggio unitario la mappa di tutti i punti che possono essere coperti (1) se tutte le telecamere sono accese.
I punti cerchiati in rosso sono coperti 2 volte, una per ogni angolazione della telecamera.
I punti contrassegnati con 0 non possono essere coperti, a differenza di quelli contrassegnati con 1. I punti caratterizzati dal 2 individuano i pilastri; al contrario del caso precedente, i pilastri non possono essere osservati dalle telecamere installate sopra di essi.
La matrice di copertura avrà 18 righe (9 telecamere per 2 angolazioni) e 121 colonne.



Può risultare utile ai fini dell'analisi dei risultati la formula che ci permette di passare dalla mappa (riga i colonna j) su Matlab alla riga della matrice di copertura Q su Python

(Q=(i-1)*11+j-1)

Vengono riportati in tabella vari test al variare di: raggio, numero di punti importanti e α.

Caso	Punti importanti	α	raggio	Punti coperti	Telecamere attive	tempo	Z	Risolvibile?
1	1 (Q=12)	0	1m	0	0	0.00s	-	no
2	1(Q=13)	0	1m	3	1 (y1)	0.00s	1	si
3	1(Q=13)	0.9	1m	27	9	0.02s	-14,40	si
4	1(Q=13)	1	1m	36	18	0.02s	-17	si
5	4(Q=13,16,95,92)	0	1m	12	4	0.00s	4	si
6	4(Q=13,16,95,92)	0	1.5m	20	4	0.01s	4	si
7	4(Q=13,16,95,92)	0.19	1.5m	20	4	0.01s	0,96	si
8	4(Q=13,16,95,92)	0.21	1.5m	45	9	0.02s	0,39	si
9	4(Q=13,16,95,92)	0.34	1.5m	72	18	0.02s	-5,12	si
10	4(Q=13,16,95,92)	1	1.5m	72	18	0.01s	-50,68	si
11	1 (Q=12)	0	16.5m	65	1 (y5)	0.02s	1	si
12	1 (Q=12)	10^{-5}	16.5m	98	1 (y7)	0.02s	1	si
13	1 (Q=12)	0.1	16.5m	121	2 (y9 e y12)	0.02s	-10	si

Nel caso 1, il problema risulta irrisolvibile poiché il punto Q=12 non rientra nell'area di copertura delle telecamere. Tuttavia, questo problema è stato risolto nel caso 11,12,13 mediante l'aumento del raggio di copertura delle telecamere a 16.5m (il problema sarebbe stato risolto anche con raggio pari a 1.5m). Per quanto riguarda il caso 2, la telecamera y_1 viene inizialmente posizionata con α=0. Con α=0.9, vengono posizionate tutte le telecamere che coprono 2 punti aggiuntivi, ossia le telecamere che garantiscono copertura ai 180° superiori. Con α=1, tutte le telecamere che coprono almeno un punto aggiuntivo vengono posizionate; vengono posizionate anche quelle che coprono i 180° inferiori.

3.3 Esempio 3

Nel terzo caso studio, sono stati posizionati ostacoli in posizioni casuali, limitando la visuale delle telecamere su alcuni punti. Nel caso analizzato, si è ipotizzato che, nonostante la presenza di ostacoli, il punto potesse comunque essere raggiunto da altre telecamere. Inoltre, si è ipotizzato che la presenza di un ostacolo non impedisca alla telecamera di osservare punti situati alle spalle di tale ostacolo.
La dimensione della mappa è stata ingrandita, conservando una struttura simile a quella illustrata nel caso 2, con un lato della mappa pari a 32 (1024 punti osservabili). Il numero di punti in cui è possibile posizionare le telecamere sono 100 (200 telecamere con le due possibili angolature), il raggio di ciascuna telecamera è stato inizialmente fissato a 3,5 metri, consentendo l'osservazione di massimo 21 punti per ogni telecamera posizionata.
Nel primo test si è ipotizzato di voler osservare solo il punto in alto a sinistra della nostra mappa, imponendo il parametro α=0. L'aspettativa è che l'algoritmo risolutore posizioni una sola telecamera con angolo=1 (rivolta verso l'alto).

```
# Creazione di un vettore di  zeri
s = np.zeros(numero_colonne)

s[0]=1
```

Di seguito si riporta l'algoritmo per la creazione degli ostacoli, con valutazione quantitativa dei punti osservabili senza ostacoli e con ostacoli.

```
numero_di_punti_osservabili_senza_ostacoli=0;   %con possibile ridondanza
for i =1:telecamere_su_lato*telecamere_su_lato*2
        for j =1:lato*lato
            numero_di_punti_osservabili_senza_ostacoli=numero_di_punti_osservabili_senza_ostacoli+C(i,j);
        end
end

%inseriamo randomicamente degli ostacoli che riducono la visibilita di cira il 10% ((lato*lato*2*telecamere_su_lato*telecamere_su_lato)/10)
for j =1:(lato*lato*2*telecamere_su_lato*telecamere_su_lato)/10
        C(randi([1,telecamere_su_lato*telecamere_su_lato*2]),randi([1,lato*lato]))=0;
end


numero_di_punti_con_ostacoli=0; %con possibile ridondanza
for i =1:telecamere_su_lato*telecamere_su_lato*2
        for j =1:lato*lato
            numero_di_punti_con_ostacoli=numero_di_punti_con_ostacoli+C(i,j);
        end
end

writematrix(C, "TEST3.txt")
```

```
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 203323 rows and 589 columns
Presolve time: 0.17s
Presolved: 2501 rows, 635 columns, 6632 nonzeros
Variable types: 0 continuous, 635 integer (635 binary)
Found heuristic solution: objective 1.0000000

Explored 0 nodes (0 simplex iterations) in 0.24 seconds (0.15 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 1

Optimal solution found (tolerance 1.00e-04)
Best objective 1.000000000000e+00, best bound 1.000000000000e+00, gap 0.0000%
y_1 = 1
y_2 = 0
```

Il modello risolutivo ha impiegato 0.24 secondi per la risoluzione, ha posizionato come ci aspettavamo 1 sola telecamera che copre 15 punti (1 importante e 14 generali).
Pertanto, si ottiene:
numero di posizioni osservate: 15
numero di telecamere posizionate: 1
numero di punti importanti: 1

```
# Creazione di un vettore di  zeri
s = np.zeros(numero_colonne)

# Genera un array casuale con valori tra 0 e 1
random_index = np.random.choice(numero_colonne, size=100, replace=False)

# Imposta gli elementi corrispondenti agli indici generati a 1
s[random_index] = 1
```

Si è notato che all'aumentare dei punti importanti posizionati casualmente, non è sempre stato possibile trovare una soluzione ammissibile al nostro problema, a causa della presenza degli ostacoli. Per risolvere questo tipo di problema, è necessario aumentare la ridondanza dei sensori: lo stesso punto deve essere coperto da più telecamere. Ciò può essere ottenuto aumentando il valore del raggio o l'angolo di copertura delle telecamere. Nel caso analizzato con raggio uguale a 3.5 m e 180° di angolo di visuale, la presenza di ostacoli non permette di vedere in nessun modo 3 punti.

Un punto non osservabile nella matrice di copertura avrà lungo la rispettiva colonna tutti zeri:

```
for i=1:lato*lato
W(i)=max(C(:,i))
end

numero_di_punti_non_osservabili=lato*lato-sum(W)
```

Vengono riportati in tabella vari test al variare di: K (numero dei punti importanti), raggio e α.

Caso	K	Numero di tentativi per generare una soluzione	α	raggio	Punti coperti	Telecamere attive	tempo	Z
1	10	1	0.1	3.5m	896	54	0.42 sec	-34.6
2	10	1	0.6	3.5m	1004	77	0.59 sec	-519
3	10	1	0.3	3.5m	971	67	0.40 sec	-221.3
4	100	2	0	3.5m	668	46	0.33 sec	46.0
5	100	1	0.1	3.5m	898	55	0.29 sec	-24.8
6	100	1	0.5	3.5m	1003	79	0.62 sec	-38.1
7	0	1	0.5	3.5m	1005	80	0.74 seco	-433
8	500	7	0.1	3.5m	976	74	0.51 sec	26.4
9	500	1	0.1	4.5m	997	46	1.16 sec	-3.7
10	500	1	0.1	48m	1022	3	0.07 sec	-49.2

Si vede come aumentando il raggio (quindi la ridondanza) nella prova 9 è bastata una sola simulazione per trovare una soluzione ammissibile; al contrario, nel caso 8 sono state necessarie sette simulazioni prima di poter trovare una soluzione ammissibile.

CAPITOLO 4. DESCRIZONE E CALCOLO DI UB E LB

4.1 Upper Bound

In un problema di copertura, l'upper bound rappresenta un limite superiore o una stima massima della soluzione ottima che si sta cercando di minimizzare. In termini più semplici, è un valore che indica quanto la soluzione ottima potrebbe al massimo essere grande. L'upper bound in un problema a minimizzare è ammissibile, e deve avvicinarsi il più possibile alla funzione obiettivo.

```
Y=zeros(1,telecamere*2);
X=zeros(1,lato*lato);
vettore_dei_gia_presi=zeros(1,lato*lato);
C_nuova=C;
for j=1:lato*lato
    if(s(j)==1)
        boolean=0;
        for i=1:telecamere*2
            if (C(i,j)==1 && boolean==0)
                boolean=1;
                vettore_dei_gia_presi=C(i,:);
                Y(i)=1;
                X=X+C(i,:);
                for jj=1:lato*lato
                      if(vettore_dei_gia_presi(jj)==1)
                        for ii=1:telecamere*2
                            C_nuova(ii,jj)=0;
                         end
                      end
               end
            end
        end
    end
end
```

Per calcolare l'Upper Bound, iniziamo con un processo iterativo nel quale posizioniamo le prime telecamere che coprono i punti indicati dal vettore s. Successivamente, creiamo una nuova matrice C (di copertura), azzerando tutte le colonne relative ai punti già coperti. Questo passo ci consente di valutare l'inserimento di nuove telecamere nell'algoritmo successivo, variando il parametro α.

```
%PARAMETRO ALPHA
ALPHA=0.5; %2 in piu accendo
for i=1:telecamere*2
    if(sum(C_nuova(i,:))>=1/ALPHA)
        boolean=0;
        for j=1:lato*lato
            if (C(i,j)==1 && boolean==0)
                boolean=1;
                vettore_dei_gia_presi=C(i,:);
                 Y(i)=1;
                X=X+C(i,:);
                for jj=1:lato*lato
                      if(vettore_dei_gia_presi(jj)==1)
                        for ii=1:telecamere*2
                            C_nuova(ii,jj)=0;
                         end
                      end
                end
            end
        end
    end
end
```

Attraverso un processo iterativo, viene valutata la somma di tutti i punti aggiuntivi che possono essere controllati mediante il posizionamento di una nuova telecamera. La relazione tra il numero di punti aggiuntivi e α è determinata dalla condizione 1/α. Vengono posizionate tutte le telecamere che coprono N* punti aggiuntivi, le rispettive colonne nella nuova matrice C vengono azzerate.

```
for kuk=1:lato*lato
if(X(kuk)>1)
    X(kuk)=1
end
end
%FUNZIONE OBBIETTIVO
Secondo_termine_J=0;
for i=1:lato*lato
    Secondo_termine_J=Secondo_termine_J+(1-s(i))*X(i)*ALPHA
end
J=sum(Y)-Secondo_termine_J

writematrix(C, "TEST2.txt")
```

Nella matrice X sono presenti il numero di volte in cui ogni punto della mappa è controllato dalle telecamere. Ai fini della funzione obiettivo, tale numero non è considerato, pertanto è stata effettuata una conversione binaria della matrice X ponendo a 1 tutti gli elementi che sono controllati.

4.2 Lower Bound

Il lower bound (limite inferiore) di un problema di minimizzazione rappresenta il valore più basso che la funzione obiettivo può assumere. Tuttavia, è importante notare che questo valore teorico può anche essere raggiunto solo da soluzioni non realistiche o non praticabili nel contesto del problema. Il lower bound potrebbe corrispondere a una soluzione che viola i vincoli o le restrizioni imposte dal problema, rendendola inaccettabile dal punto di vista pratico.

```
#vincoli
for j in R:
    if s[j]==1: #aggiungo per forza i punti importanti
        lu=1;
        #mod.addConstr(gp.quicksum(A[i,j]*y[i] for i in L) >= 1)
    else:     #altrimenti, il punto è generale e la sua appartenza alla soluzione dipende da alpha
        mod.addConstr(gp.quicksum(A[i,j] * y[i] for i in L) >= x[j])
    for i in L:
        mod.addConstr((A[i,j]*y[i])<=x[j])  #se posiziono la telecamera devo segnare i punti coperti
```

Per ottenere il lower bound, è stato eliminato il vincolo che imponeva che i importanti dovessero necessariamente essere coperti da una telecamera risolvendo il problema SCP associato.

4.3 Test calcolando gli UB e LB

Cominciamo considerando i test su UB e LB nel terzo esempio (mappa 32x32 con presenza di ostacoli e raggio 3.5m)

Caso	K	α	Z UB	Z	Z LB	X UB	X	X LB	Y UP	Y	Y LB
1	100	0	71	43	0	847	673	0	71	43	0
2	100	0.1	-3.00	-24.8	-27.9	900	898	899	77	55	52
3	100	0.5	-351.5	-381	-384	1005	1003	1002	110	79	76
4	10	0.5	-396	-430	-430	1010	1005	1005	114	77	77
5	0	0.5	-399.5	-433	-433	1007	1005	1005	114	80	80
6	500	0.1	69.6	26.4	-4.4	1014	976	714	121	74	17

Si nota che all'aumentare di α la soluzione ottima Z e il Z LB tendono a coincidere.,
L'algoritmo risolutore del LB impiega in media il 25% del tempo in meno rispetto l'algoritmo di risoluzione originale, questo comportamento cambia al variare del numero di punti importanti.

I test su UB e LB sul secondo esempio (mappa 11x11 senza presenza di ostacoli e raggio 3.5m)

K	α	Z UB	Z	Z LB	X UB	X	X LB	Y UP	Y	Y LB
5	0.21	-13.52	-15,52	-15,52	117	117	117	10	8	8

I test su UB e LB sul primo esempio

K	α	Z UB	Z	Z LB	X UB	X	X LB	Y UP	Y	Y LB
3	0	2	2	0	6	6	0	2	2	0
3	0.334	0.994	0.994	0.994	12	12	12	4	4	4

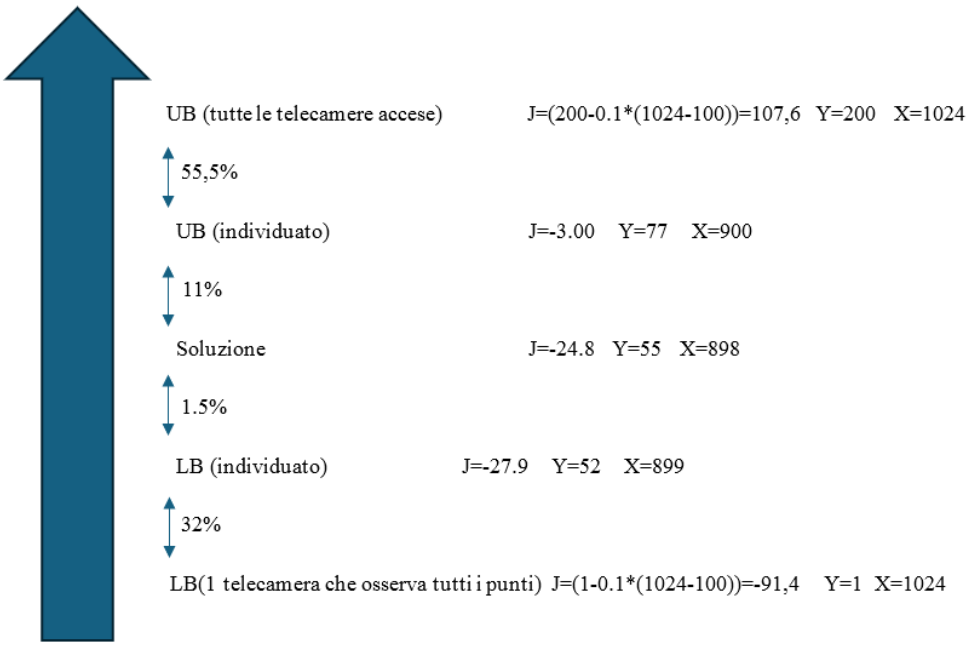
```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1
```

```
0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1
```

4.4 Valutazione dell'UB e del LB considerati

Per valutare la bontà della stima dell'upper bound e del Lower bound è possibile effettuare una comparazione con un UB e un LB di bassissima precisione ovvero:
- UB tutte le telecamere sono posizionate ed accese
- LB una sola telecamera osserva tutti i punti

Facendo riferimento all'esempio 3 (mappa 32x32 con raggio 3.5m e 2 angolazioni possibili) dove i punti importanti sono 100 e α=0.1, una telecamera è posizionata per coprire punti generali solo se può vedere 10 o più punti aggiuntivi.



L’algoritmo di upper bound può essere ulteriormente migliorato. Infatti, nella prima parte, quando si valutano le telecamere che devono coprire i punti importanti, si è aggiunta alla soluzione la prima telecamera che copre un punto importante; per migliorare l’algoritmo si potrebbe pensare di realizzare un algoritmo di upper bound che aggiunge la telecamera che copre il punto importante ed il maggior numero di punti generici.

```
for j=1:lato*lato
    if(s(j)==1)
        boolean=0;
        for i=1:telecamere*2
            if (C(i,j)==1 && boolean==0)
                boolean=1;
                vettore_dei_gia_presi=C(i,:);
                Y(i)=1;
                X=X+C(i,:);
```