

Corso di Laurea in Ingegneria Elettronica e Informatica

A.A. 2018/19



# Simulazione di algoritmi di navigazione robotica: **VISIBILITY GRAPH E TRAPEZOIDAL DECOMPOSITION**

Relatore:  
**Tullio Facchinetti**

Relazione discussa in sede di  
esame finale dal candidato:  
**Riccardo Doveri**

# SOMMARIO



UNIVERSITÀ  
DI PAVIA

- Contesto
- Obiettivi
- Strumenti utilizzati
- Creazione mappa
- Gestione segmenti
- Algoritmi implementati
- Struttura del codice
- Esempi
- Conclusione

The screenshot shows the GitHub repository page for 'PythonRobotics' by AtsushiSakai. The repository has 390 watches, 8.3k stars, and 2.6k forks. It includes a 'Join GitHub today' banner and a list of Python sample codes for robotics algorithms. The repository statistics show 1,420 commits, 1 branch, 0 packages, 1 release, 50 contributors, and MIT license. A table of recent commits is displayed below.

Commit	Message	Time
AtsushiSakai Merge pull request #291 from sldai/issue_285	Latest commit 06438d4 11 days ago	
.github	Delete greetings.yml	2 months ago
AerialNavigation	Merge pull request #259 from goktug97/close_on_key	2 months ago
ArmNavigation	Merge pull request #259 from goktug97/close_on_key	2 months ago
Bipedal	add comment for stopping the simulation	2 months ago
Introduction	start writing introcution doc	13 months ago
InvertedPendulumCart	inverted pendulum mpc control is added	14 days ago
Localization	#fix 281 and doc clean up	19 days ago
Mapping	Merge pull request #259 from goktug97/close_on_key	2 months ago
PathPlanning	fix RRT rewiring	12 days ago

# CONTESTO



UNIVERSITÀ  
DI PAVIA

- ☐ PythonRobotics
- ☐ Progetto Open Source
- ☐ Navigazione robotica
- ☐ Simulazioni autocontenute



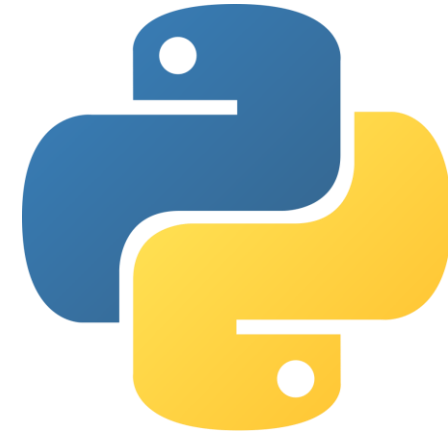
Un sistema di navigazione **autonomo** è un sistema che permette ad un robot di spostarsi verso una destinazione all'interno di un ambiente in cui sono presenti ostacoli, **senza il controllo di un operatore**.

# OBIETTIVI

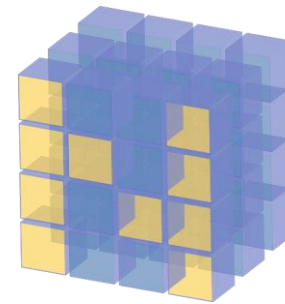
- ☐ Possibilità di lettura di una qualsiasi mappa geometrica
- ☐ Creazione di un grafo rappresentante le adiacenze tra i nodi della mappa
- ☐ Calcolo del percorso migliore per raggiungere il nodo di goal partendo da start
- ☐ Visualizzazione grafica dell'intera simulazione

# STRUMENTI UTILIZZATI

- ❑ Linguaggio di programmazione: **Python**
  - Ottime librerie per operazioni matematiche
  - Leggibilità del codice
- ❑ Librerie: **Math, Matplotlib, Numpy**
  - Grafica semplice ed intuitiva



**matplotlib**



**NumPy**

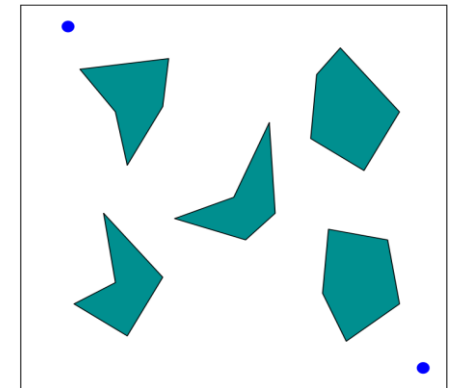
# CREAZIONE DELLA MAPPA



UNIVERSITÀ  
DI PAVIA

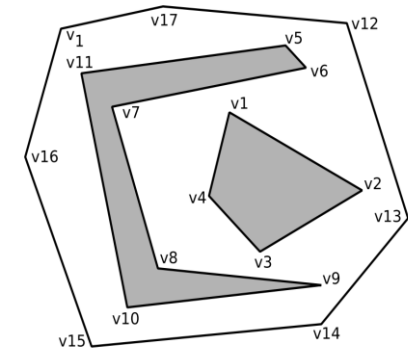
Una mappa è una **struttura dati** che rappresenta l'ambiente in cui un robot si muove.

- ☐ Mappa geometrica
- ☐ Sistema di **coordinate**
- ☐ Ostacoli **poligonali** (concavi e/o convessi)



Sono state differenziate due tipologie di mappe geometriche:

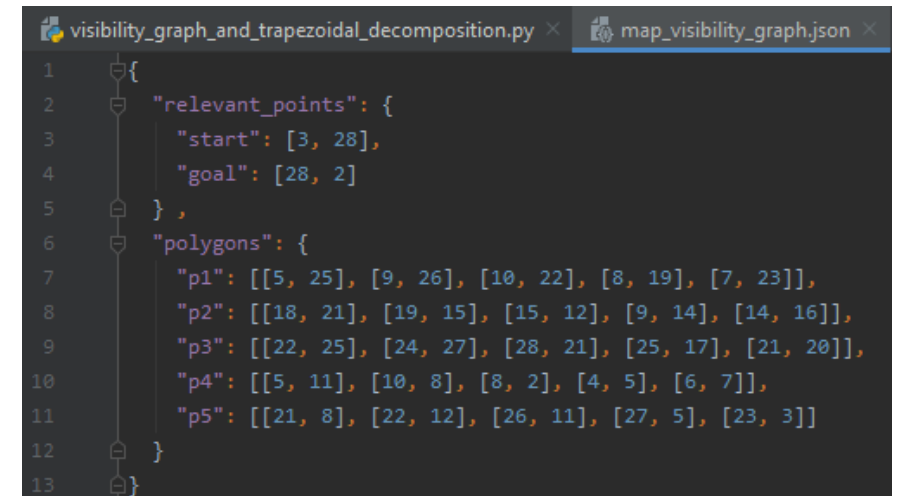
- ☐ Illimitate
- ☐ Limitate da un ostacolo esterno



# CREAZIONE DELLA MAPPA

- ❑ Mappa di default
- ❑ Possibilità di lettura della mappa da file json
  - Semplicità di utilizzo
  - Stesso formato per entrambi i tipi di mappa (in caso di mappe limitate l'ostacolo esterno viene inserito come primo)

Per differenziare a livello logico un ostacolo esterno è utilizzato un attributo booleano «isDelimiter»



```
1 {  
2   "relevant_points": {  
3     "start": [3, 28],  
4     "goal": [28, 2]  
5   } ,  
6   "polygons": {  
7     "p1": [[5, 25], [9, 26], [10, 22], [8, 19], [7, 23]],  
8     "p2": [[18, 21], [19, 15], [15, 12], [9, 14], [14, 16]],  
9     "p3": [[22, 25], [24, 27], [28, 21], [25, 17], [21, 20]],  
10    "p4": [[5, 11], [10, 8], [8, 2], [4, 5], [6, 7]],  
11    "p5": [[21, 8], [22, 12], [26, 11], [27, 5], [23, 3]]  
12  }  
13 }
```




## Segmenti:

- Collegamenti tra nodi
- Divisione dello spazio



Intersezioni tra segmenti calcolate analiticamente da un apposita funzione

## Problemi:

- Intersezioni troppo vicine agli estremi  `numpy.isclose()`
- Segmenti particolari (diagonali interne ed esterne)  Teorema di Jordan
- Intersezioni con segmenti verticali (coefficiente angolare infinito)  Funzione apposita



# ALGORITMI IMPLEMENTATI



UNIVERSITÀ  
DI PAVIA

Per la creazione di grafi, nodi e adiacenze:

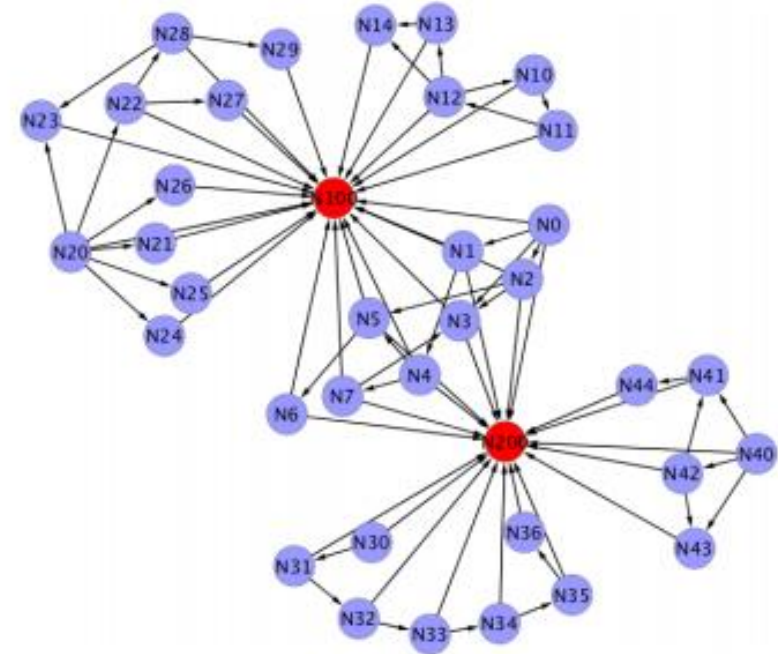
- ☐ Visibility Graph
- ☐ Trapezoidal Decomposition (mappe limitate)

Per il calcolo del percorso migliore:

- ☐ A\*

Possibili applicazioni:

- ☐ Esplorazione
- ☐ Sorveglianza
- ☐ Manifattura



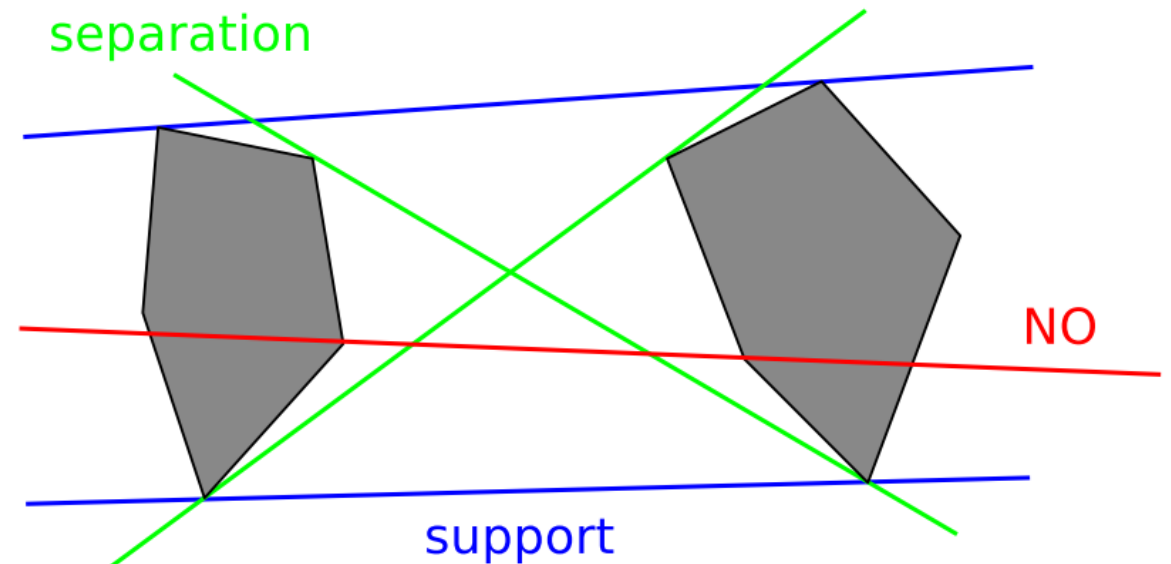
# VISIBILITY GRAPH



UNIVERSITÀ  
DI PAVIA

I nodi sono rappresentati dai punti di **start**, **goal** e da tutti i **vertici** dei poligoni.

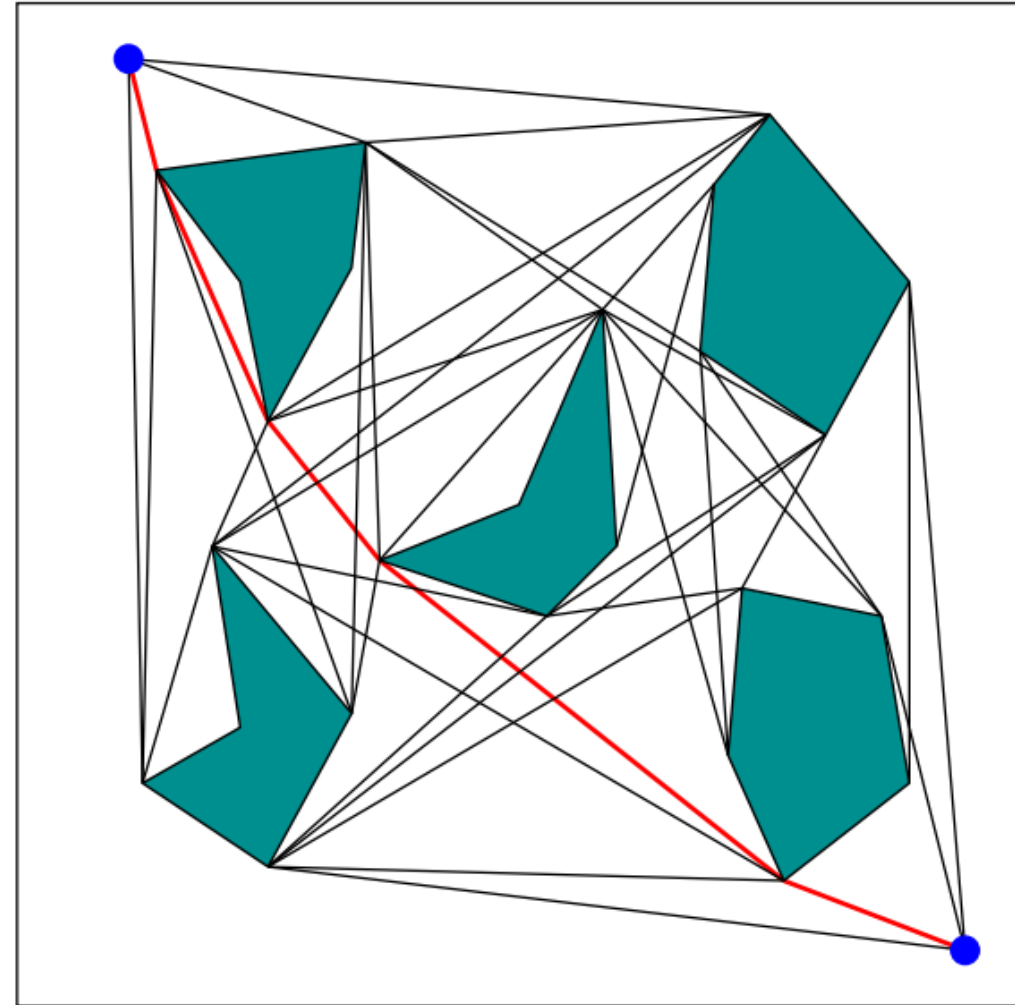
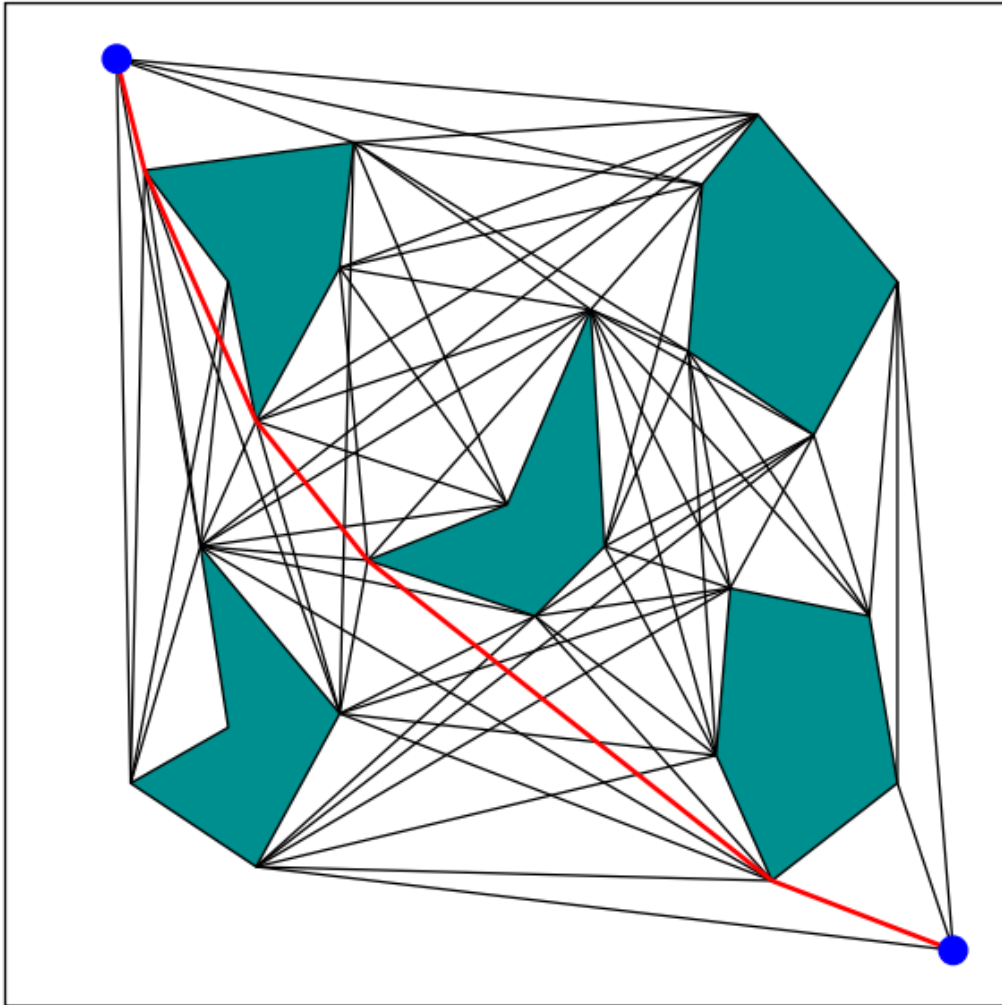
- ❑ **Completo**: considero **tutti** i collegamenti possibili tra nodi, purché non vi siano intersezioni con gli ostacoli
- ❑ **Ridotto**: elimino i collegamenti ridondanti e non necessari considerando solamente i segmenti di **supporto** e di **separazione**



# VISIBILITY GRAPH



UNIVERSITÀ  
DI PAVIA



# TRAPEZOIDAL DECOMPOSITION



UNIVERSITÀ  
DI PAVIA

Assunzione: lo spazio libero è **delimitato** da un **poligono**.

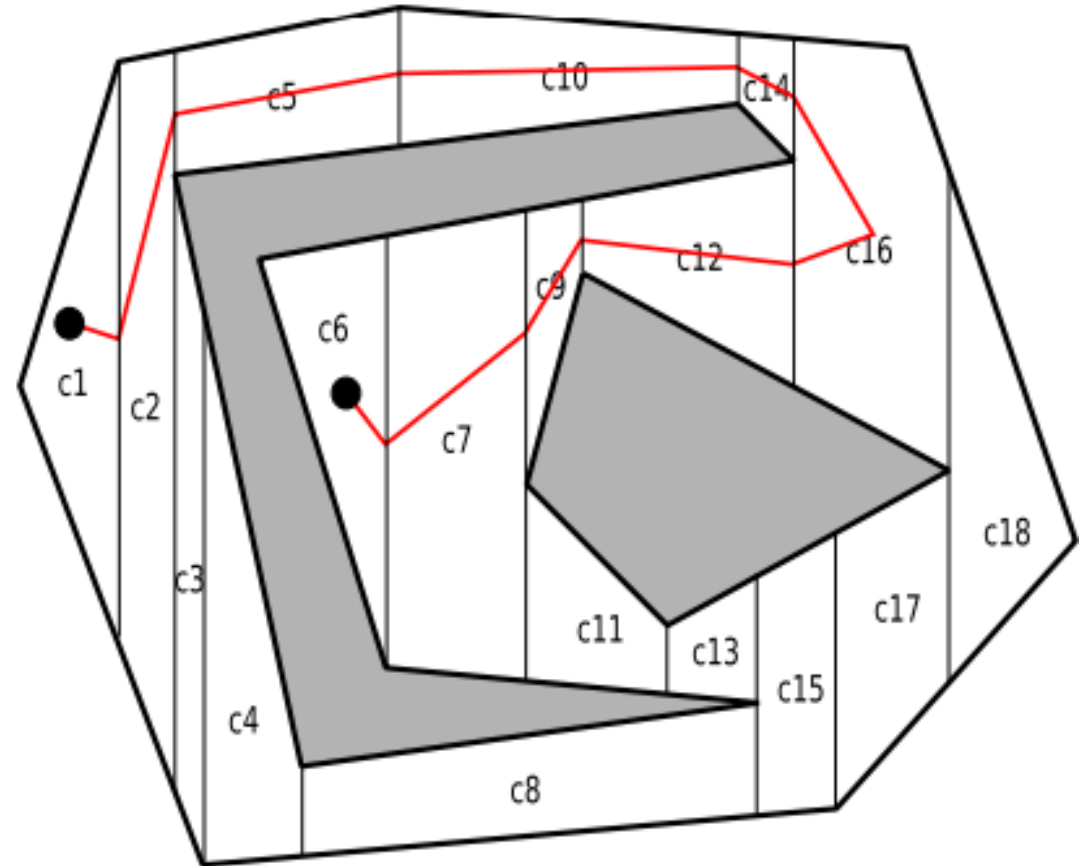
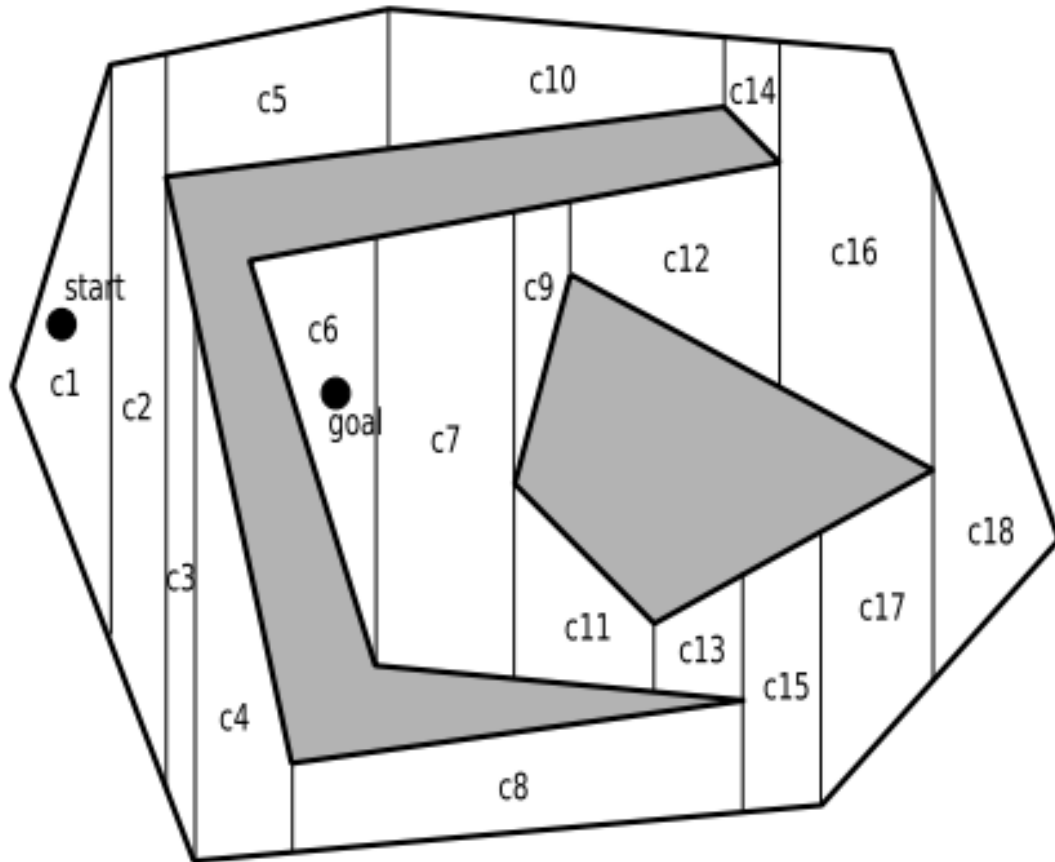
- ❑ Da ogni **vertice** vengono generati due **segmenti** (se possibile) la cui estensione si ferma al primo ostacolo incontrato
- ❑ I segmenti possono essere generati in qualsiasi direzione, purché siano **paralleli**
- ❑ Spazio diviso in **trapezi** e **triangoli** (trapezi con un lato nullo)
- ❑ Identificazione delle aree contenenti i punti di start e di goal

I nodi sono rappresentati dalle **aree** ottenute.

# TRAPEZOIDAL DECOMPOSITION



UNIVERSITÀ  
DI PAVIA

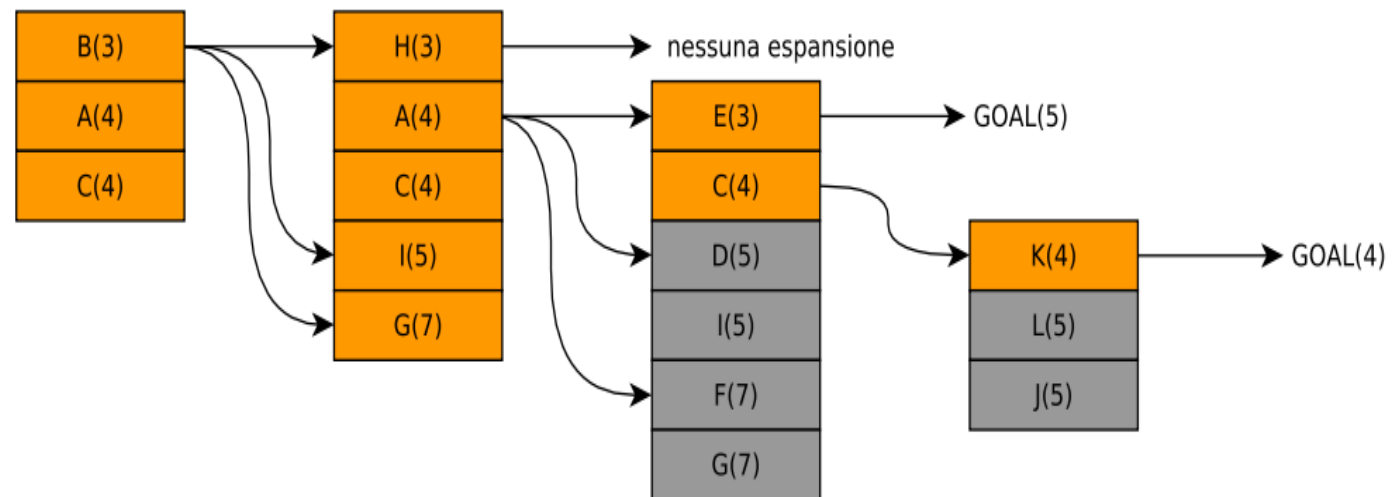
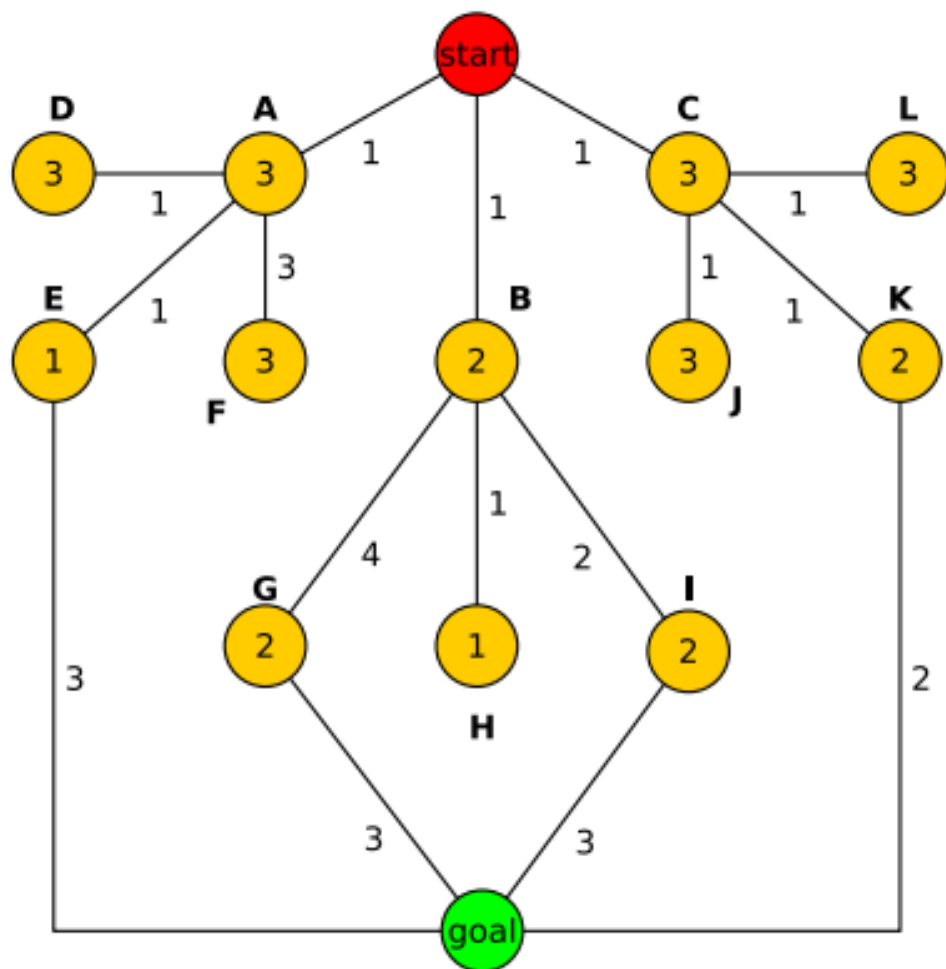


# A\*

- ❑ Sviluppato per trovare un percorso in un grafo
- ❑ Basato sulla conoscenza della posizione di goal
- ❑ Utilizza una ricerca euristica
- ❑ L'euristica è utilizzata per scegliere la direzione del movimento
- ❑ Prende in considerazione la distanza tra la posizione corrente, il punto di start e il punto di goal

Come distanza è stata considerata la distanza geometrica tra le coordinate dei nodi.

# A\*



# A\*

L'algoritmo implementato richiede una lista di nodi aventi:

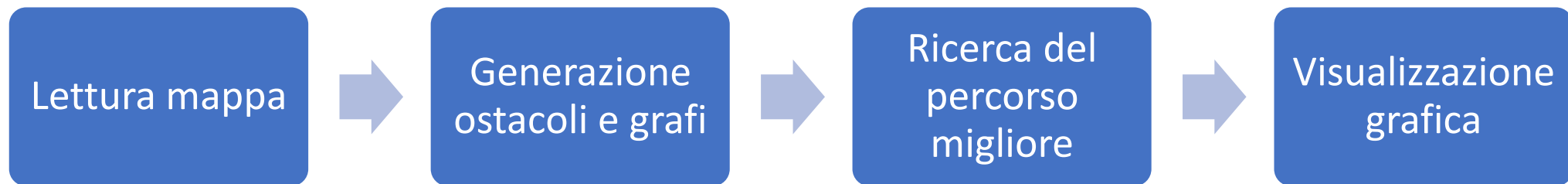
- Coordinate
- Lista di nodi adiacenti con rispettive distanze
- Distanza da goal

Metodo «move\_to\_best\_neigh» implementato per il passaggio al nodo successivo, eseguito finché non viene trovato il percorso migliore.



# STRUTTURA DEL CODICE

- ❑ Gran parte del codice in **comune**: figure geometriche, gestione intersezioni, mappa
- ❑ **A\*** **comune** a entrambi gli algoritmi, applicabile inoltre a qualsiasi grafo
- ❑ Differenze: creazione Visibility Graph e Trapezoidal Decomposition

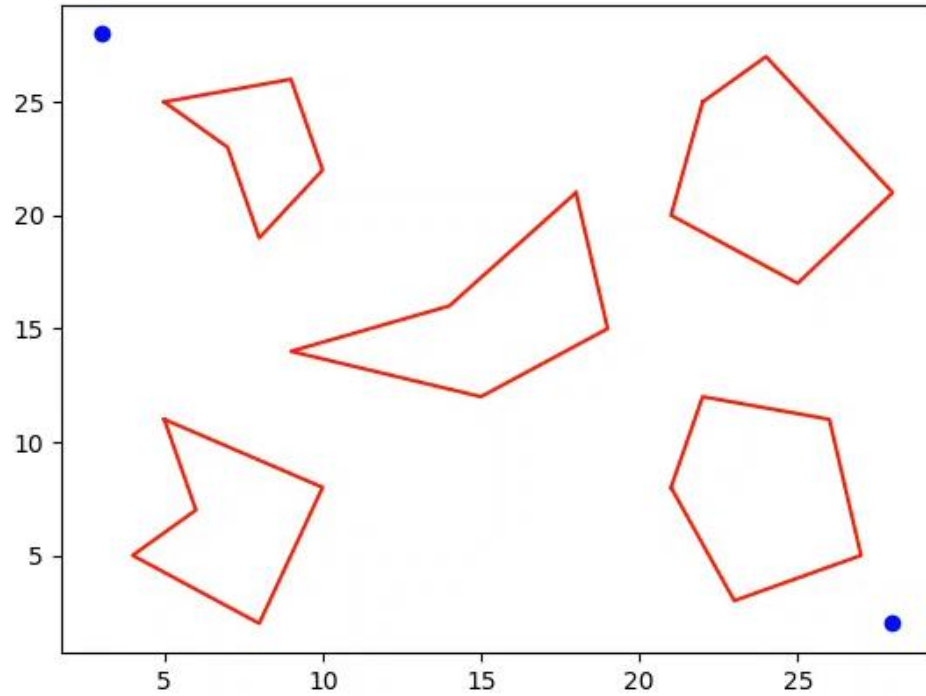


# VISIBILITY GRAPH

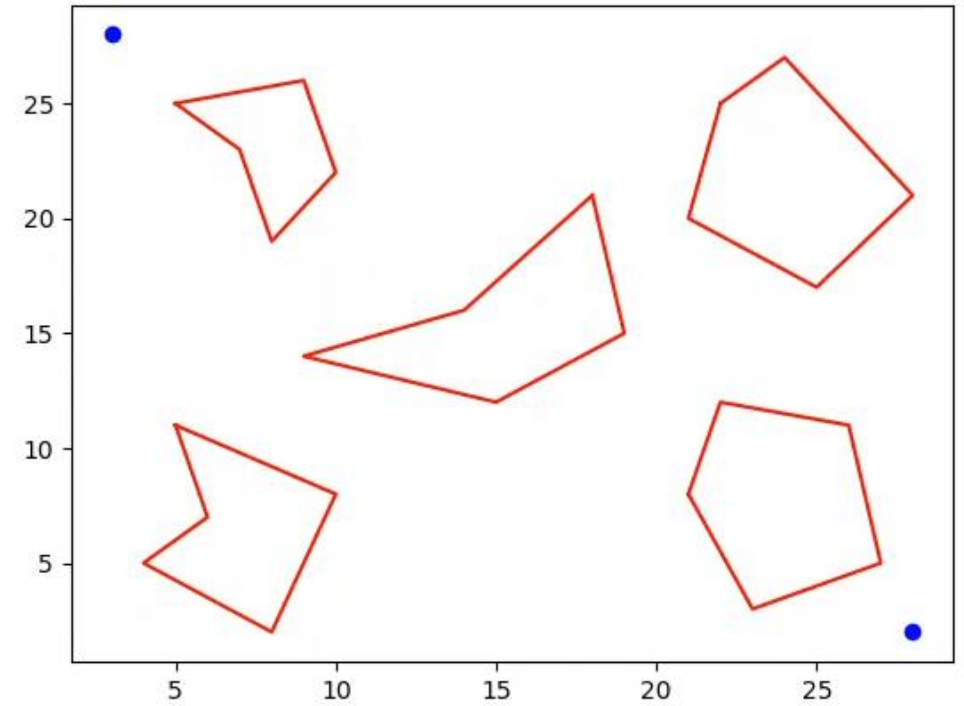


UNIVERSITÀ  
DI PAVIA

Completo



Ridotto

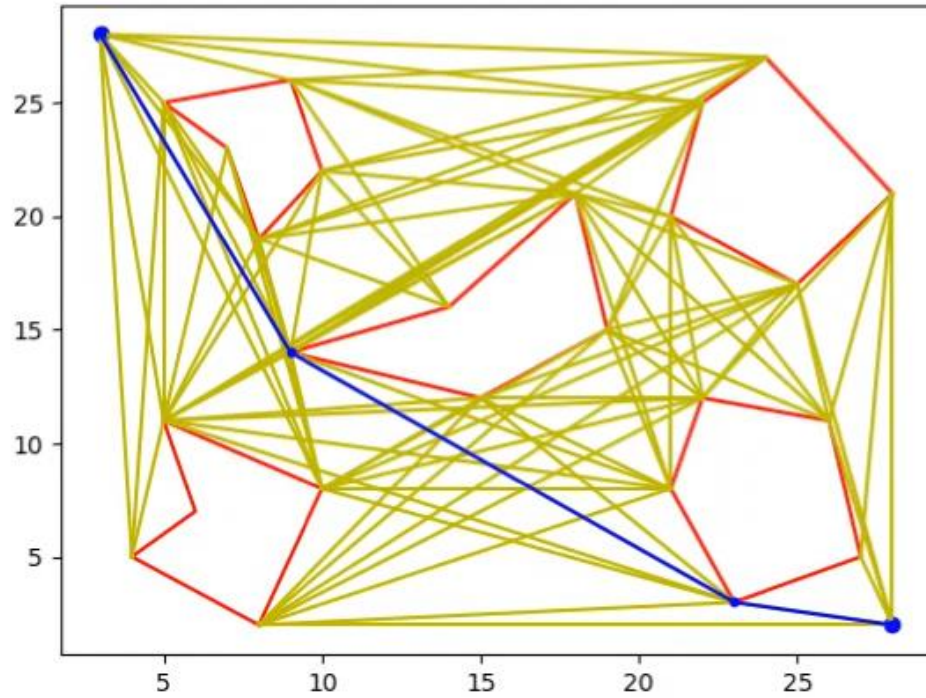


# VISIBILITY GRAPH

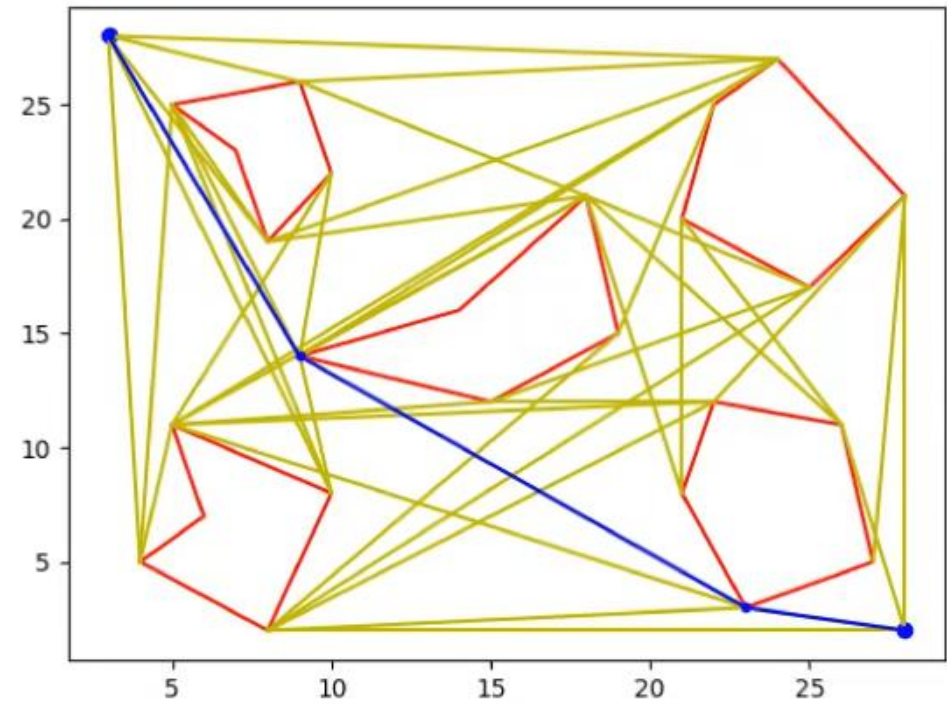


UNIVERSITÀ  
DI PAVIA

Completo



Ridotto

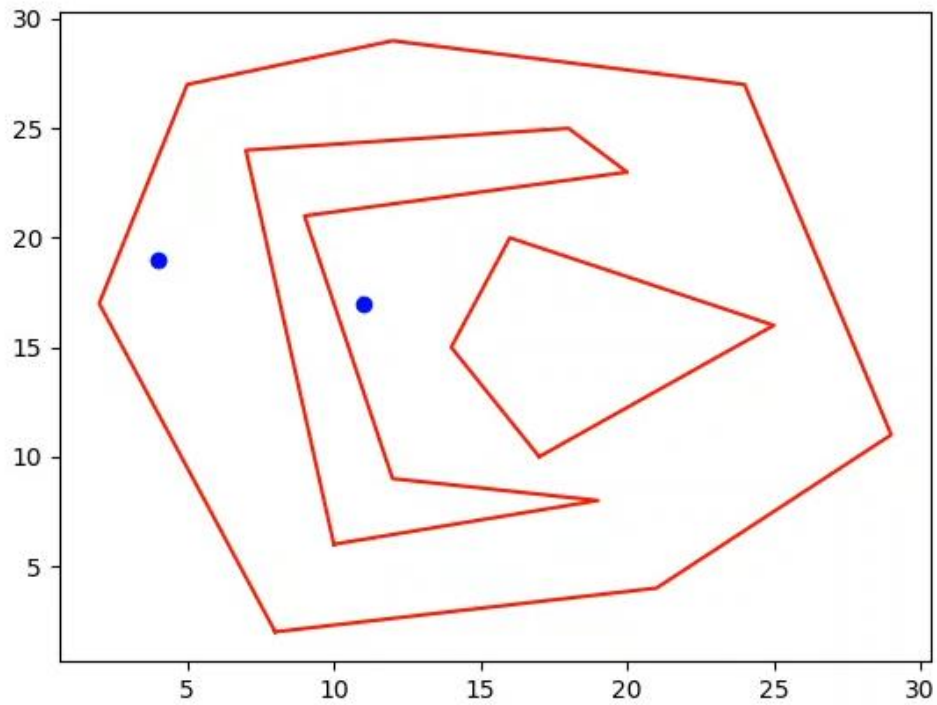


# TRAPEZOIDAL DECOMPOSITION

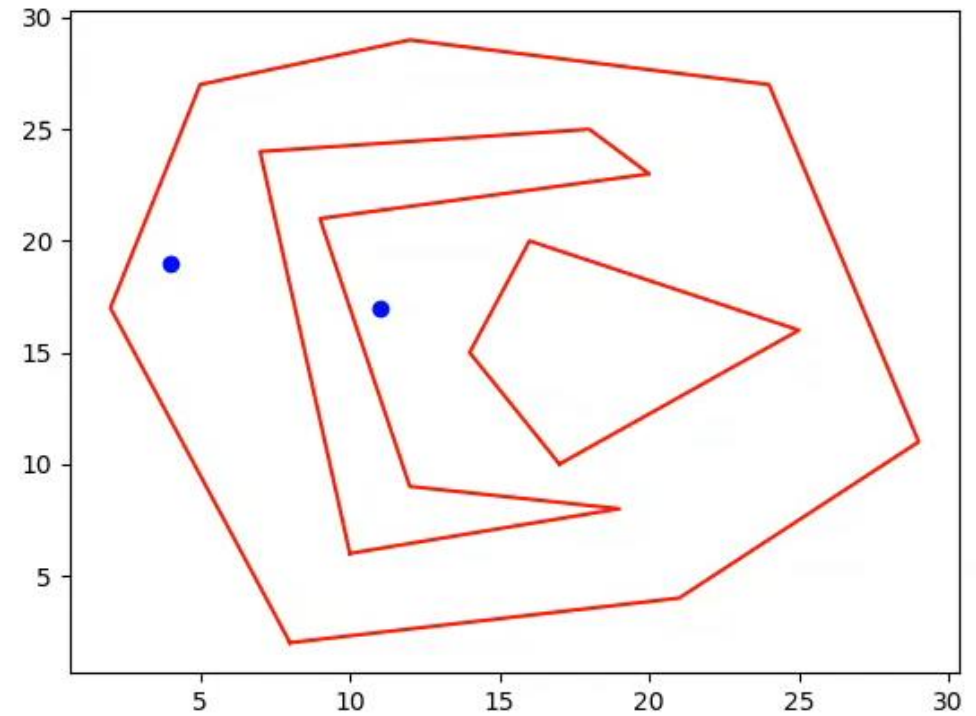


UNIVERSITÀ  
DI PAVIA

$M = 0$



$M = 0.5$

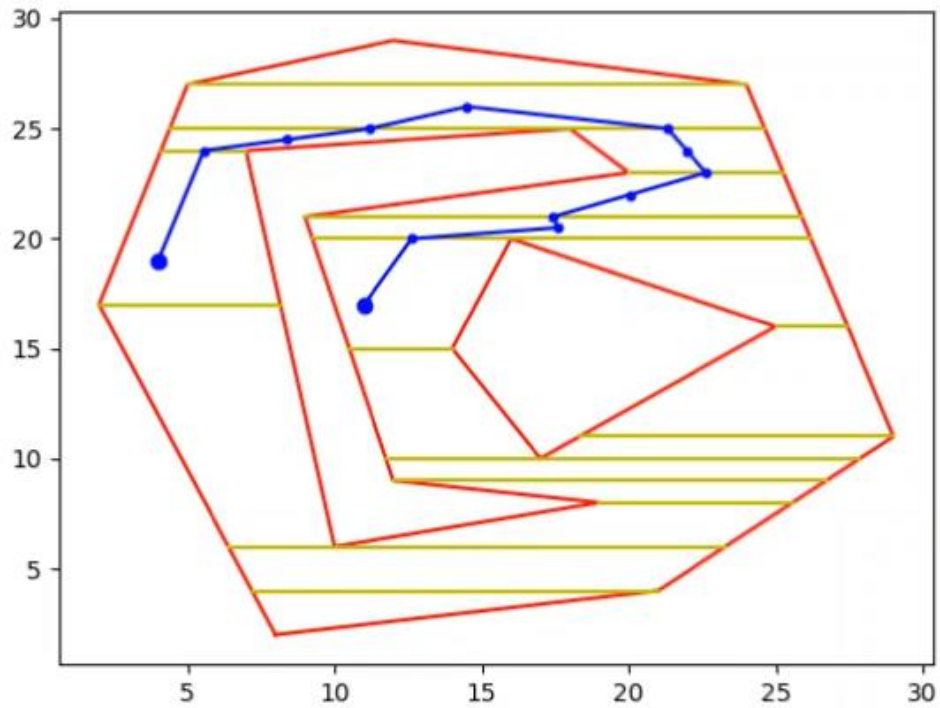


# TRAPEZOIDAL DECOMPOSITION

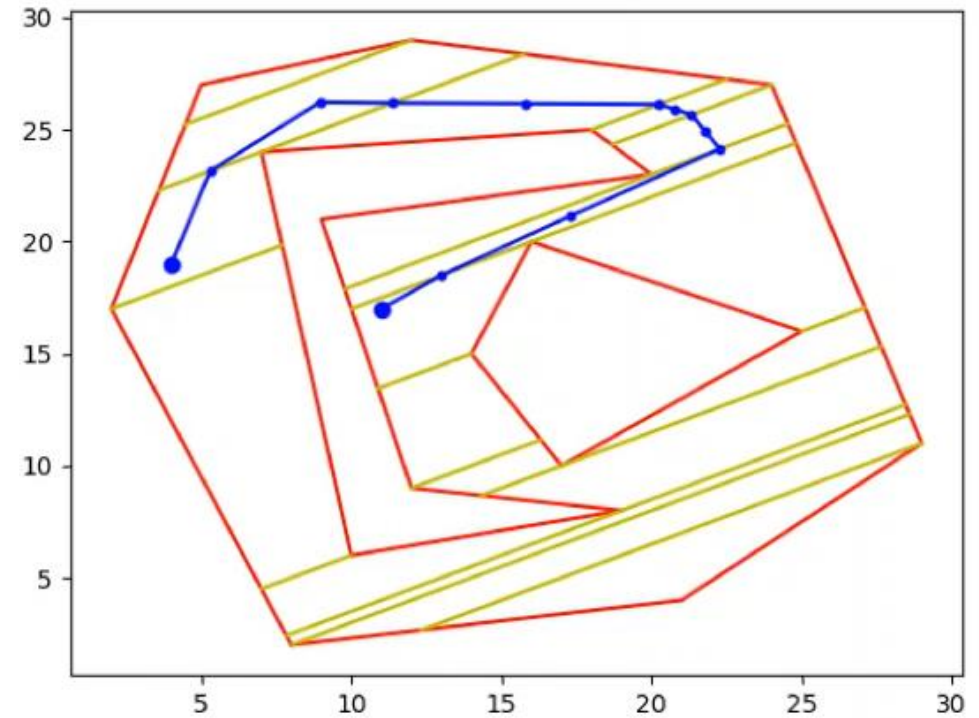


UNIVERSITÀ  
DI PAVIA

$M = 0$



$M = 0.5$



# CONCLUSIONI

- ❑ Sistema di lettura della mappa da file **json**
- ❑ Sono state realizzate le simulazioni degli algoritmi Visibility Graph e Trapezoidal Decomposition che verranno **integrate nell'ambiente di PythonRobotics**
- ❑ È stato implementato l'algoritmo  $A^*$  **applicabile a un qualsiasi grafo** in un sistema di coordinate cartesiane
- ❑ Sono stati seguiti i **paradigmi della OOP** per semplificare l'integrazione di nuovi algoritmi che facciano uso di parti comuni
- ❑ È stata realizzata una **grafica semplice ed intuitiva** per una migliore comprensione degli algoritmi