

# SAAC

Sistema di Assistenza per Amministrazioni Comunali  
un progetto di Ingegneria della Conoscenza a.a. 2022-2023



Gianluca Laera  
mat. 718556  
repository github

## Sommario

Progetto di ingegneria della conoscenza che propone l'implementazione di un sistema multiagente intelligente per la gestione efficiente delle operazioni in un edificio. Il sistema è composto da due agenti cooperanti: uno responsabile della pianificazione dei percorsi tra le stanze e l'altro incaricato di monitorare lo stato dell'impianto elettrico dell'edificio. L'agente di pianificazione dei percorsi utilizza algoritmi basati sulla conoscenza per determinare i cammini più efficienti, tenendo conto della disponibilità di risorse come ascensori. D'altro canto, l'agente diagnostico si occupa del monitoraggio costante dell'impianto elettrico, rilevando anomalie e proponendo soluzioni tempestive basate su regole di conoscenza.

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Note di Progetto . . . . .	1
1.2	Argomenti trattati . . . . .	1
1.3	Strumenti utilizzati . . . . .	1
1.4	L'edificio . . . . .	2
<b>2</b>	<b>Agente Diagnostico</b>	<b>4</b>
2.1	Monitoraggio . . . . .	4
2.2	Analisi e Diagnosi . . . . .	4
2.2.1	La Base di conoscenza . . . . .	4
2.2.2	Fatti . . . . .	5
2.2.3	Regole . . . . .	6
2.3	Avvisi e Suggerimenti . . . . .	8
<b>3</b>	<b>Ricerca di un cammino efficiente</b>	<b>9</b>
3.1	La Base di Conoscenza . . . . .	9
3.1.1	Fatti . . . . .	9
3.1.2	Regole . . . . .	9
3.2	Il problema di ricerca . . . . .	10
3.2.1	Definizione dell'euristica . . . . .	10
3.2.2	L'algoritmo di Ricerca . . . . .	11
3.3	La costruzione del Grafo di ricerca . . . . .	11
<b>4</b>	<b>Funzionamento</b>	<b>13</b>
<b>5</b>	<b>Conclusione</b>	<b>14</b>

# 1 Introduzione

SAAC è un sistema multiagente che si pone come obiettivo quello di semplificare ed automatizzare alcune delle operazioni che possono essere effettuate all'interno di un edificio, in questo caso si tratterà di un edificio di amministrazione comunale.

I diversi agenti intelligenti del sistema cooperano tra loro attraverso la comunicazione, infatti tramite lo scambio di informazioni e competenze, coordinano le proprie azioni per raggiungere un obiettivo comune.

Nello specifico le funzionalità attualmente offerte dal sistema sono le seguenti:

- Ricerca del percorso più efficiente tra due posizioni all'interno dell'edificio (sezione 3)
- Manutenzione e controllo del sistema elettrico dell'edificio (sezione 2)

Le due funzionalità sopra introdotte sono svolte tutte da un agente diverso, ciascun agente coopera con l'altro. L'agente che si occupa della ricerca del percorso coopera con quello che esegue i controlli di manutenzione durante la creazione del percorso, ad esempio se un ascensore non è funzionante verrà notificato dal secondo agente, quindi l'agente che ricerca il cammino non considererà quell'ascensore nel percorso.

La cooperazione tra agenti può essere un processo complesso, ma può portare a risultati molto efficaci, occorre però trovare un modo per coordinare le loro azioni in modo efficiente ed efficace, oltre che garantire che condividano tra loro informazioni in modo sicuro e affidabile e soprattutto un metodo di gestione dei conflitti che possono generarsi.

Nelle sezioni seguenti si procede a descrivere ciascuno degli agenti implementati in maniera dettagliata

## 1.1 Note di Progetto

Questo è un progetto svolto nell'ambito del corso di Ingegneria della Conoscenza A.A. 2022-2023, essendo quindi per lo più con intento didattico sono stati fatti dei compromessi.

L'obiettivo è soprattutto dimostrare di aver appreso gli argomenti trattati durante il corso, pertanto si è scelto di soffermarsi più in dettaglio su quest'ultimi e sulle conseguenze che ne derivano, dunque si è dedicato meno spazio alle implementazioni a livello di codice.

Nello specifico del codice sono state fatte scelte a livello implementativo che favoriscono la leggibilità e la comprensione piuttosto che l'efficienza (pur mantenendo una complessità asintotica accettabile).

Gli ambienti descritti nel progetto non corrispondono ad ambienti reali, poiché per motivi legali non è stato possibile recuperare planimetrie esatte, dunque sono stati creati da zero cercando di mantenere comunque una certa verosimiglianza.

## 1.2 Argomenti trattati

- Spazi di Stati e Ricerca Soluzioni
- Rappresentazione e Ragionamento Proposizionale
- Rappresentazione e Ragionamento Relazionale

## 1.3 Strumenti utilizzati

Gli agenti sono stati scritti interamente in python, essi sfruttano delle basi di conoscenza scritte in Prolog. Per permettere la comunicazione tra python e Prolog si è utilizzato il modulo pyswip, che permette di interrogare basi di conoscenza Prolog direttamente da python.

Per implementare la costruzione del grafo di ricerca e i conseguenti algoritmi di ricerca si è sfruttata la libreria "aipython" fornita dal libro di testo [1], nello specifico si è fatto uso del file searchProblem.py per costruire il grafo di ricerca, mentre dei file searchGeneric.py, searchMPP.py per implementare gli algoritmi di ricerca scelti.

## 1.4 L'edificio

Come già detto, la planimetria è stata inventata da zero cercando di mantenere una certa verosimiglianza.

Quest'ultima è poi stata codificata in una base di conoscenza (KB) di fatti e regole scritta in Prolog.

Tutte le stanze dell'edificio sono collegate tramite corridoi, per spostarsi tra i piani è possibili utilizzare sia scale che ascensore.

La planimetria è stata trattata in modo ambivalente:

- sia come **Grafo**: questo è utile per l'applicazione degli algoritmi di ricerca di un cammino su grafo
- sia come **Griglia**: poiché dotata di coordinate, rende più semplice ed immediato la definizione di un'Euristica per l'algoritmo di ricerca.

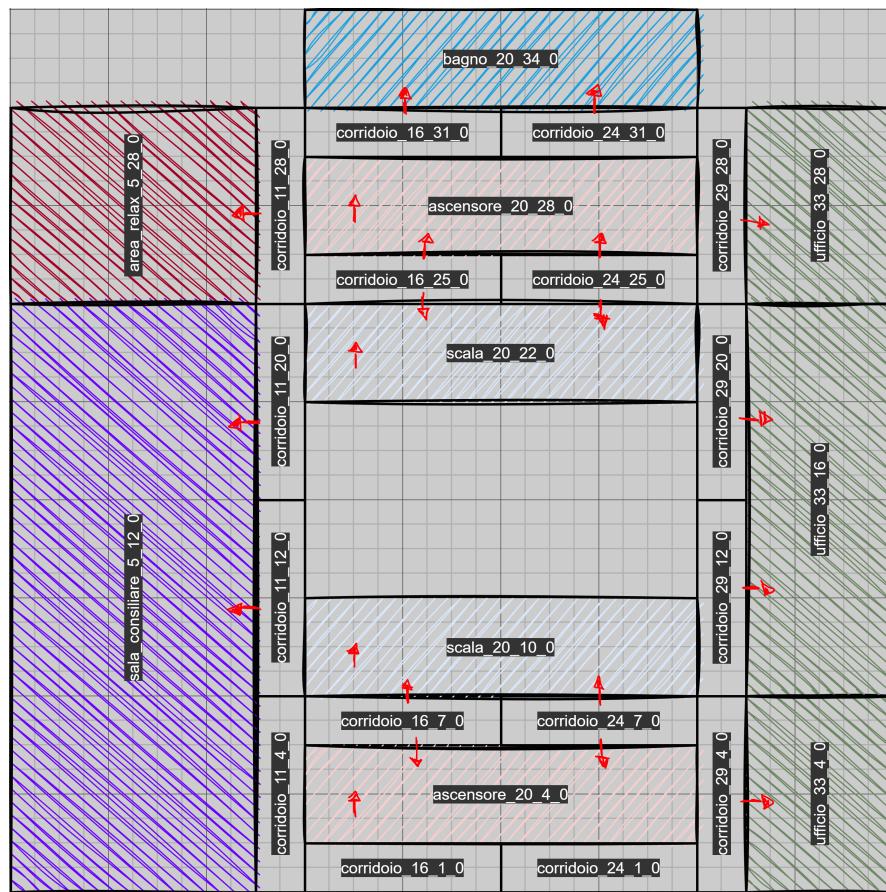


Figura 1: Planimetria piano terra

Come si vede dalle Figure 1,2 la nomenclatura delle diverse stanze segue una forma standard: nomeStanza\_X\_Y\_piano, dove X e Y sono le coordinate del centro del rettangolo che identifica la stanza (il sistema di coordinate ha origine nel vertice in basso a sinistra), il piano indica appunto il piano di appartenenza (piano terra = 0). La descrizione del piano può essere vista come un'ulteriore coordinata sull'asse Z, questo sarà importante quando si andrà a calcolare l'euristica per l'algoritmo di ricerca per due stanze su piani diversi.

- I corridoi sono percorribili in tutte le direzioni;
- le stanze sono accessibili solo dai corridoi che hanno la freccia verso la stanza.

Per modellare il cambio di piano si è scelto di "duplicare" i nodi ascensore e scale, ovvero per ogni piano esistono questi nodi, sebbene nella realtà rappresenterebbero entità uniche. Dunque ogni nodo

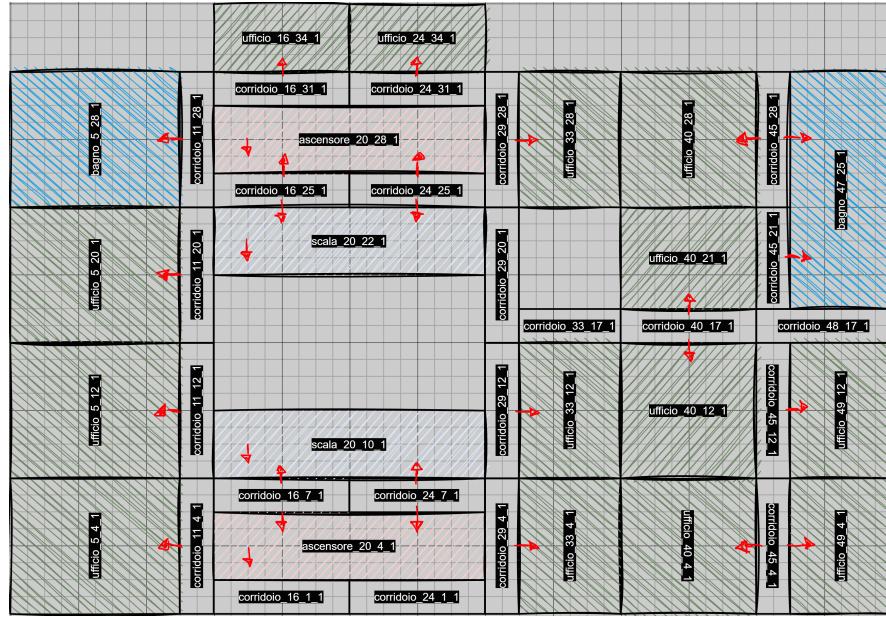


Figura 2: Planimetria primo piano

di questo tipo ha un arco allo stesso nodo del piano diverso. Quindi per raggiungere un nodo ad un altro piano si è costretti a passare per questi nodi. Questa scelta ha comportato la necessità di trovare una soluzione particolare per la gestione dei malfunzionamenti riscontrati dall'agente diagnostico su uno di questi nodi particolari. Infatti, sebbene siano duplicati nella rappresentazione prodotta, nella realtà sono unici, quindi i dispositivi interni ad essi sono gli stessi, pertanto se c'è un guasto ad "ascensore\_20\_4\_1" deve esserci lo stesso guasto ad "ascensore\_20\_4\_0".

## 2 Agente Diagnostico

Un agente diagnostico per l'impianto elettrico (EE) di un edificio dovrebbe avere diverse funzionalità per monitorare e diagnosticare eventuali problemi nel sistema. Le funzionalità che si è scelto di integrare poiché considerate utili includono:

- **Monitoraggio** in tempo reale dei dati provenienti dai sensori e dai dispositivi di misura dell'impianto elettrico.
- **Analisi** dei dati per rilevare eventuali anomalie o malfunzionamenti.
- **Diagnosi** dei problemi utilizzando tecniche di intelligenza artificiale e algoritmi di ragionamento basati sulla conoscenza.
- **Generazione di avvisi** e notifiche in caso di problemi rilevati.
- **Fornire suggerimenti e raccomandazioni** per la risoluzione dei problemi.

### 2.1 Monitoraggio

Il monitoraggio in tempo reale dello stato dell'EE viene effettuato attraverso sensori, per motivi di test, si è scelto di simulare il comportamento di questi sensori tramite degli script Python che generano dati in modo del tutto casuale o seguendo un certo modello.

Dunque a run-time l'agente raccoglie i dati dai sensori e asserisce i fatti riguardanti questi dati nella base di conoscenza, alla pari di vere e proprie osservazioni sull'ambiente. L'agente sfrutta come conoscenza di fondo una base di conoscenza Prolog che assiomatizza fatti riguardanti i diversi dispositivi come le loro posizioni nell'edificio, a questa quindi aggiunge le osservazioni effettuate tramite sensori sui diversi dispositivi.

- In particolar modo saranno raccolti dati sulla corrente che attraversa i dispositivi, sulla tensione e sul tempo di utilizzo. Saranno raccolti dati anche riguardo gli interruttori, ovvero la corrente che li attraversa, sulla base della quale si può inferire se l'interruttore è abbassato o alzato.

Ricordando la planimetria dell'edificio nelle figure 1 e 2 l'impianto elettrico è stato così organizzato:

- Ogni stanza dell'edificio ha una luce, un interruttore che la controlla e un climatizzatore
- Per le luci sono state assiomatizzate due tipologie di anomalie, la rottura della stessa oppure la suggerita sostituzione quando si è superato il tempo di vita consigliato
- Gli altri dispositivi invece, in questo caso il climatizzatore, possono essere affetti dall'anomalia del sovraccarico.
- Ogni stanza è dotata di un interruttore magnetotermico.

Le diverse diagnosi saranno approfondite nella sezione successiva.

### 2.2 Analisi e Diagnosi

#### 2.2.1 La Base di conoscenza

Per prima cosa si è individuato il **livello d'astrazione** da considerare per la rappresentazione di un EE:

- Come discusso nella sezione Note di Progetto, considerato il reale obiettivo che si vuole raggiungere, si è scelto un livello d'astrazione estremamente semplice. L'impianto elettrico viene rappresentato in modo che sia comprensibile anche ai non esperti del dominio. Attualmente sono fornite informazioni riguardo lo stato ed il funzionamento di solo 3 tipologie di dispositivi oltre che il possibile guasto che può colpirli.

– luce, interruttore e climatizzatore

Tuttavia si è mantenuto lo sviluppo modulare in modo che fosse possibile l'aggiunta di nuovi dispositivi apportando modifiche minime.

Individuata l'astrazione si è passato al processo di **concettualizzazione**, ovvero si sono mappati i simboli definiti nel processo di astrazione con gli individui e le relazioni identificati da questi simboli.

Possiamo distinguere questi individui e relazioni in 4 macro aree:

1. **Dispositivi**: individui che rappresentano vari dispositivi nell'EE, appunto interruttori e luci. Ogni dispositivo ha delle proprietà come lo stato (acceso/spento), la corrente elettrica che lo attraversa, la tensione, ecc.
2. **Località**: relazioni che indicano la stanza in cui si trova il dispositivo.
3. **Anomalie**: individui che rappresentano varie anomalie o problemi che possono verificarsi nell'impianto, come cortocircuiti, sovraccarichi, guasti dell'interruttore, ecc.
4. **Regole di Diagnosi**: regole che rappresentano la logica di diagnosi dell'agente. Ad esempio, se la corrente su un circuito supera un certo limite, allora c'è un sovraccarico.

A questo punto si è ottenuta l'**interpretazione intesa**. Dunque si passa all'**assiomatizzazione del dominio**, ovvero alla costruzione di fatti e regole che siano veri all'interno dell'interpretazione intesa, quindi alla costruzione della vera e propria base di conoscenza.

### 2.2.2 Fatti

Si analizzano alcuni dei fatti assiomatizzati nell'KB spiegandone il comportamento:

device (NomeDispositivo , Corrente , Tensione , Durata)

- Questo predicato serve a relazionare un dispositivo con la corrente che lo attraversa, la tensione, e il tempo di utilizzo. I fatti di questo tipo vengono asseriti a run-time dall'agente diagnostico, di fatto rappresentano i dati che l'agente riceve dai sensori (simulati).

switch (Switch , Corrente)

- Come per il predicato precedente anche questo viene asserito a run-time e rappresenta i dati appresi dai sensori sugli interruttori.

is\_light (DispositivoLuce)

- Fatto che individua il dispositivo come una luce.

is\_heater (DispositivoClimatizzatore)

- Fatto che individua il dispositivo come un climatizzatore.

controlled\_by (Luce , Switch)

- Relazione che lega una luce al rispettivo interruttore che l'accende o la spegne.

location (Device , Room)

- Relazione che lega un dispositivo alla stanza in cui si trova

magnetothermal\_on (Room)

- Fatto che rappresenta se il magnetotermico nella stanza Room è attivo (vedere sezione seguente)

### 2.2.3 Regole

Si analizzano alcun delle regole assiomatizzate nell KB spiegandone il comportamento e le scelte prese:

---

```
diagnosis(Device, broken_light) :-  
    is_light(Device),      % verifico che è una luce  
    \+is_on(Device),       % verifico che sia spenta  
    controlled_by(Device, Switch), % trovo l'interruttore che la controlla  
    is_switch_on(Switch), % verifico che l'interruttore sia abbassato (acceso)
```

---

- Questa regola diagnostica un problema alla luce. Occorre però effettuare una premessa, si è partito da un'assunzione: "Tutte le luci dell'impianto sono luci led di nuova generazione", infatti questa tipologia di luci possiede al proprio interno dei sensori che impediscono il fluire di corrente in caso di malfunzionamenti. Per questo motivo per capire se la luce è rotta basta verificare che essa non sia in funzione, controllare che l'interruttore ad essa collegata sia abbassato, verificare quindi che arrivi corrente all'interruttore. Si può escludere che il problema sia dovuto al cavo che collega luce e interruttore, in quanto sarebbe un grave problema di progettazione, perché vorrebbe dire che si è messo un cavo la cui portata di corrente nominale è minore di quella effettiva. Oltre tutto grazie ai sensori delle luci di cui sopra, ci si può permettere di trascurare il deterioramento di un cavo poiché questo avviene soprattutto a causa di condizioni anomale come una sovraccorrente, che, come detto in precedenza, è difficile che accada poiché nel caso di questo tipo di problema i sensori della lampadina bloccherebbero immediatamente la corrente. Pertanto l'unica possibile diagnosi è che la luce sia rotta.

---

```
% Il dispositivo è acceso se attraversato da corrente  
is_on(Device) :-  
    device(Device, Current, _, _),  
    Current > 0.  
  
% L'interruttore è acceso se attraversato da corrente  
is_switch_on(Switch) :-  
    switch(Switch, Current),  
    Current > 0
```

---

- Verifica se il dispositivo è acceso o l'interruttore è abbassato, vedendo se è attraversato da corrente. In un ambiente realistico la corrente non può essere mai 0, poiché lo strumento utilizzato per misurare la corrente (amperometro) ha anche esso una resistenza interna. Questo significa che, anche se non c'è corrente nel circuito da misurare, una piccola corrente scorrerà comunque attraverso l'amperometro, la corrente di fondo. Tuttavia ai fini dell'astrazione qui decisa si è ritenuto trascurabile questo fenomeno, anche perché si tratta di una corrente estremamente bassa.

---

```
% Sostituzione delle lampadine dopo 10000 ore di utilizzo  
diagnosis(Light, replace_light) :-  
    is_light(Light),  
    device(Light, _, _, Hours),  
    Hours > 10000.
```

---

- Regola che individua se è opportuno sostituire la lampadina. Generalmente le lampadine hanno una durata garantita stabilita dal produttore, la possibilità di poter contare le ore di utilizzo potrebbe essere utile per sostituire una lampadina prima che smetta di funzionare per evitare perdite di tempo qualora desse problemi durante l'orario lavorativo. Oppure potrebbe essere utile per un'estensione futura, come ad esempio un agente che sceglie se ordinare o meno delle nuove lampadine sulla base della durata residua di quelle attuali.
- 

```
% Diagnostica un sovraccarico quindi rimuove la corrente nella stanza
diagnosis(Device, overload) :-
    is_heater(Device), %verifica che il dispositivo non sia una luce
    device(Device,Current,_,_),
    Current > 20, %valore in Ampere della corrente massima che può fluire
    location(Device, Room), % trova la stanza in cui si trova il dispositivo in sovraccarico
    active_magnetothermal(Room). % attiva il magnetotermico per rimuovere immediatamente la corrente
    → nella stanza
```

---

- Anche in questo caso occorre fare una premessa. L'interruttore magnetotermico è un dispositivo di sicurezza utilizzato negli impianti elettrici per proteggere la linea elettrica da un eventuale sovraccorrente elettrica. Questo può verificarsi nel caso di un sovraccarico, l'anomalia che si è deciso di modellare. Il sovraccarico si verifica quando viene richiesta più corrente di quella che il circuito elettrico è in grado di sopportare, portando quindi al surriscaldamento degli isolanti, che potrebbero sciogliersi e causare un incendio. In questo caso interviene il magnetotermico e interrompe la corrente. Si è ipotizzato che ogni stanza fosse dotata di un magnetotermico, questa è una situazione molto comune negli ospedali dove in caso di malfunzionamenti non è ammesso di rimuovere la corrente in tutto l'edificio, quindi viene rimossa solo nella stanza interessata.
  - Alla luce di quanto detto la regola qui sopra diagnostica un sovraccarico quando un dispositivo (attualmente solo un climatizzatore) supera una certa soglia di corrente, quindi fa "scattare" il magnetotermico della stanza.
- 

```
% Attiva il magnetotermico che stacca la corrente nella stanza
active_magnetothermal(Room) :-
    location(Light, Room), %individua la luce nella stanza
    is_light(Light), %verifica sia una luce
    controlled_by(Light, Switch), %trova l'interruttore che la controlla
    location(Device, Room), %trova il climatizzatore nella stanza
    is_heater(Device), %verifica che sia un climatizzatore
    % rimuove la corrente
    retract(device(Light, _, _, Hlight)),
    retract(switch(Switch, _)),
    retract(device(Device, _, _, Hheater)),
    assertz(device(Light, 0, 220, Hlight)),
    assertz(device(Device, 0, 220, Hheater)),
    assertz(switch(Switch, 0)),
    assertz(magnetothermal_on(Room)). %asserisce che il magnetotermico nella stanza è scattato
```

---

- La regola di attivazione del magnetotermico per prima cosa recupera le informazioni riguardanti tutti i dispositivi nella stanza (luce, interruttore, climatizzatore), quindi rimuove i fatti che li riguardano tramite il predicato retract/1 e asserisce i nuovi fatti con il predicato assertz/1, fatti che non sono altro che i precedenti ma con la corrente impostata a 0. Questo è stato realizzato per simulare l'intervento del magnetotermico ed il suo "staccare" la corrente

---

```
:-
    :- dynamic
        device/4,
        controlled_by/2,
        switch/2,
        magnetothermal_on/1.
```

---

- Questa regola sfrutta il PredicateIndicator `dynamic` di Prolog che rende dinamici i predicati ad esso associati così che possano essere asseriti o rimossi a run-time.

Le regole rimanenti sono state omesse perché servono più per le operazioni interne al sistema, ma sono comunque scritte nella sezione in cui verrà mostrato tutto il codice della base di conoscenza.

## 2.3 Avvisi e Suggerimenti

Definita la base di conoscenza l'agente diagnostico inizia ad operare a pieno regime.

Per le anomalie codificate, il sistema genera avvisi stampati a schermo, questi includono informazioni sulla natura dell'anomalia e sulla posizione nell'impianto. Sulla base dell'anomalia, il sistema fornisce suggerimenti per risolvere il problema, che possono essere azioni correttive specifiche, la sostituzione di componenti difettosi o altre misure preventive.

Eventuali anomalie che possono risultare in problematiche per l'agente che ricerca i cammini, vengono prontamente notificate a quest'ultimo. Ad esempio uno dei casi considerati è quello che prevede un malfunzionamento della stanza "ascensore" che porta a far "scattare" il magnetotermico, quando questo accade l'ascensore ovviamente non è utilizzabile poiché non c'è corrente, questo viene prontamente notificato all'agente che calcola il percorso. Quest'ultimo prenderà atto dell'ascensore non funzionante e non lo considererà nella costruzione del grafo di ricerca.

### 3 Ricerca di un cammino efficiente

#### 3.1 La Base di Conoscenza

Sviluppata la planimetria dell'edificio, sono stati individuati individui e relazioni che meglio descrivono la rappresentazione intesta, dunque si è proseguito con la codifica della KB in Prolog proprio come fatto per l'agente diagnostico.

Si è scelto di tenere fatti e regole in due file distinti prettamente per motivi organizzativi, infatti, dato il loro esoso numero, separarli ha permesso di allestire meglio la base di conoscenza rendendone più semplice la gestione. Inoltre in questo modo si è favorita una progettazione più modulare, che ha garantito una più semplice aggiunta o modifica di parti senza influire su tutto il resto.

##### 3.1.1 Fatti

Vengono riportati di seguito un esempio per ciascun tipo di fatto inserito nella KB

---

```
bath(bagno_20_34_0).
elevator(ascensore_20_28_0).
floor(area_relax_5_28_0,0).
hallway(corradoio_11_12_0).
office(ufficio_16_34_1).
misc(sala_consiliare_5_12_0).
stair(scala_20_10_0).
relax(area_relax_5_28_0).
coordinates(area_relax_5_28_0,5,28,0).
edge(area_relax_5_28_0,corradoio_11_28_0).
```

---

Listing 1: Tipologia di Fatti inseriti nella KB

Come si può vedere i fatti asseriti sono principalmente la codifica della planimetria, identificano le varie costanti che individuano i nomi delle stanze, a questi poi si aggiungono quelli che delineano la rappresentazione sotto forma di grafo, tramite appunto la definizione degli archi e la relazione che lega ogni stanza alle proprie coordinate.

##### 3.1.2 Regole

---

```
% X è un nodo del grafo se: X è un bagno OR X è un ascensore Or...
node(X) :- 
    bath(X);
    elevator(X);
    hallway(X);
    misc(X);
    office(X);
    relax(X);
    stair(X).

% Distanza euclidea tra 2 stanze (nodi)
distance(P1,P2,Distance) :-
    node(P1),
    node(P2),
    coordinates(P1, X1, Y1, Z1),
    coordinates(P2, X2, Y2, Z2),
    X is X2 - X1,
    Y is Y2 - Y1,
    Z is Z2 - Z1,
    Distance is sqrt(X * X + Y * Y + Z * Z).
```

---

Le prima regola rappresenta cosa è effettivamente un nodo mentre la seconda regola serve a calcolare la distanza euclidea tra due nodi, questa è l'euristica scelta per l'algoritmo utilizzato per la ricerca del cammino.

## 3.2 Il problema di ricerca

La ricerca dei cammini è stata implementata utilizzando la libreria **aipython** messa a disposizione dal libro di testo [1].

La classe `SearchProblemSAAC` estende la classe `Search_problem.from_explicit_graph` dal file `searchProblem.py`, per adattare al meglio la costruzione del grafo di ricerca al problema di riferimento.

Visto il numero limitato di nodi del caso di studio e considerato che comunque le stanze di un edificio sono relativamente limitate si è scelto di utilizzare un algoritmo che operasse su un grafo esplicito e non che lo costruisse nel proseguire della propria esecuzione.

Inoltre si è scelto di operare con un algoritmo di ricerca informata che sfruttasse un'euristica per riconoscere i nodi più promettenti per raggiungere l'obiettivo.

### 3.2.1 Definizione dell'euristica

La funzione euristica è una funzione che associa ad ogni nodo del grafo un numero reale positivo, che non è nient'altro che la stima del costo minima di un percorso da un nodo "n" fino a un nodo "obiettivo".

L'euristica svolge un ruolo cruciale in un algoritmo di ricerca informata ed è fondamentale per migliorarne l'efficienza e la capacità di trovare soluzioni ottimali o vicine all'ottimo in tempi ragionevoli.

In particolare un'euristica deve essere **ammissibile** e **consistente**:

- **ammissibile**: se sottostima il costo, ovvero la funzione è sempre minore o tutt'al più uguale del costo minima del percorso da un nodo "n" a un nodo "obiettivo"
- **consistente**: se è una funzione non negativa su un nodo n che soddisfa il seguente vincolo:

$$h(n) \leq cost(n, n') + h(n')$$

- per ogni coppia di nodi n' e n
- dove  $cost(n, n')$  è il costo del cammino minimo da n a n'

La consistenza è garantita se l'euristica soddisfa la **restrizione di monotonicità**, ovvero soddisfa il medesimo vincolo precedente per ogni arco  $\langle n, n' \rangle$ , quest'ultima è più facilmente dimostrabile in quanto dipende solo dagli archi.

In questa situazione corre in aiuto la **disuguaglianza triangolare** che specifica come la lunghezza di ogni lato di un triangolo non può essere più grande della somma delle lunghezze degli altri due lati, si può sfruttare questa relazione per dimostrare la consistenza:

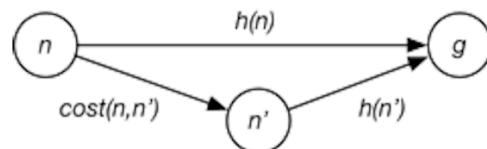


Figura 3: Disuguaglianza triangolare

- Come si è detto in precedenza il costo stimato del cammino da n a g non deve essere maggiore del costo stimato del percorso che obbliga il passaggio per n'

È possibile osservare come la **distanza euclidea** soddisfi la disuguaglianza triangolare, infatti, se la funzione di costo è quest'ultima, la funzione euristica che è la distanza più corta tra il nodo n e il nodo obiettivo soddisfa la restrizione monotona.

Vista la possibilità, all'interno del grafo definito nel problema di questo progetto, di muoversi a proprio piacimento verso qualsiasi direzione, la scelta della distanza euclidea come euristica è stata automatica.

Si è poi considerato di utilizzare due euristiche differenti a seconda che nodo di partenza e di arrivo si trovassero su piani diversi. In particolare si sarebbe usata la distanza euclidea per i nodi nello stesso piano. Mentre per i nodi su piani diversi si sarebbe seguita la tecnica di utilizzare una funzione

euristica che fosse soluzione di un problema semplificato, ovvero prima si sarebbe trovato il cammino minimo al primo nodo disponibile per raggiungere il secondo piano, poi questo si sarebbe sommato al cammino minimo per raggiungere il nodo obiettivo dal nodo del secondo piano raggiunto al passo precedente.

Tuttavia avendo codificato la planimetria in un certo modo ed avendo a disposizione anche le coordinate per identificare il piano, a chi scrive è sembrato poco proficuo definire due euristiche diverse, quando la distanza euclidea per un sistema a tre coordinate funziona sufficientemente bene.

### 3.2.2 L'algoritmo di Ricerca

Come algoritmo di ricerca si è optato per l'algoritmo A\* con potatura dei cammini (MPP).

L'algoritmo A\* sfrutta l'euristica per trovare un cammino a costo minimo, proprio in previsione all'utilizzo di questo algoritmo che si è dedicato del tempo a cercare un'euristica valida. Inoltre grazie all'ammissibilità dell'euristica questo algoritmo consente di selezionare come primo il percorso a costo minimo verso un nodo goal.

A questo algoritmo si è poi aggiunta la potatura dei cicli, per questo motivo si è cercata un'euristica che fosse anche consistente oltre che ammissibile, perché, in questo modo, si garantisce che si trovi per primo il percorso minimo verso un qualunque nodo che non sia per forza un nodo obiettivo, quindi permettendo di effettuare la potatura.

La scelta dell'algoritmo è ricaduta sull'A\* con MPP per via degli ovvi risultati più efficienti, sebbene nel caso particolare di questo progetto i nodi siano comunque troppo pochi per vedere un grosso impatto prestazionale, come si può vedere dalla figura 4. In letteratura è però ben noto quanto questo algoritmo possa essere efficiente specialmente presa un'euristica particolarmente efficace.

```
Creazione del grafo di ricerca terminata con successo
Creazione avvenuta in 0.0 secondi.

10047 paths have been expanded and 22004 paths remain in the frontier
Ricerca terminata in 0.06783771514892578 secondi.

A*Searcher:
Caminno trovato: corridoio_16_1_0 --> corridoio_11_4_0 --> corridoio_16_7_0 --> scala_20_10_0 --> scala_20_10_1 --> corridoio_24_7_1 --> corridoio_29_4_1 --> corridoio_29_12_1 --> corridoio_29_20_1 --> corridoio_33_17_1 --> corridoio_40_17_1 --> corridoio_48_17_1 --> corridoio_45_21_1 --> bagno_47_25_1
costo: 73.96499163953547

Creazione avvenuta in 0.0 secondi.

48 paths have been expanded and 35 paths remain in the frontier
Ricerca terminata in 0.0009946823126117188 secondi.

A*Searcher con MPP:
Soluzione: corridoio_16_1_0 --> corridoio_11_4_0 --> corridoio_16_7_0 --> ascensore_20_4_0 --> ascensore_20_4_1 --> corridoio_24_7_1 --> corridoio_29_4_1 --> corridoio_29_12_1 --> corridoio_29_20_1 --> corridoio_33_17_1 --> corridoio_40_17_1 --> corridoio_48_17_1 --> corridoio_45_21_1 --> bagno_47_25_1
costo: 73.96499163953547
```

Figura 4: Valutazione dell'algoritmo di ricerca A\* con e senza MPP

Si veda infatti come il primo algoritmo (senza MPP) abbia espanso 10047 cammini mentre il secondo per lo stesso cammino ne abbia espansi solo 48, questo lascia prevedere quanto ancora meglio, in proporzione, si comporterebbe su un grafo con più nodi.

### 3.3 La costruzione del Grafo di ricerca

L'agente che ricerca il cammino è modellato dalla classe PathSearchAgentAAC, che, sfruttando la libreria aipython, esegue la ricerca utilizzando l'algoritmo discusso precedentemente.

Una menzione va fatta alla costruzione del grafo di ricerca, infatti l'agente si serve della classe SearchProblemAAC che estende la classe della libreria di cui sopra Search\_problem\_from\_explicit\_graph. La prima classe elabora i dati della planimetria trasformandoli in nodi e archi, recupera il nodo di partenza e i nodi obiettivo dall'agente, quindi calcola l'euristica di tutti i nodi, a questo punto passa tutto al costruttore della superclasse che si occupa di generare il grafo esplicito vero e proprio.

I nodi del grafo sono modellati dalla classe Node che tramite interrogazioni alla KB recupera il nome del nodo, le sue coordinate e la lista dei vicini.

Come si può vedere dal codice 2, ottenuta tutta la lista dei nodi, che altro non sono che ogni stanza dell'edificio, come visto nella sezione sulla base di conoscenza, si esamina ogni nodo uno per uno e si

```

1      def __init__(self, kb: KnowledgeBase, da : ElectricalDiagnosticAgent, start, goals):
2          """
3              Elabora le informazioni della planimetria in nodi e costruisce gli archi tra i nodi se sono
4              disponibili
5                  :param kb: riferimento alla base di conoscenza
6                  :param da: riferimento all'agente diagnostico
7                  :param start: nodo di partenza del percorso
8                  :param goals: nodo di arrivo del percorso
9                  """
10             self.kb = kb
11             hmap_heuristic = {}
12             arcs = []
13
14             # per prima cosa creo l'insieme dei nodi
15             nodes = self.get_set_of_nodes()
16             node_labels = self.get_set_of_node_names()
17
18             # calcolo l'euristica per ogni nodo
19             for node in nodes:
20                 # Se il nodo è un ascensore
21                 if "ascensore" in node.get_label():
22                     # Se il magnetotermico è attivo l'ascensore non funziona
23                     if da.check_magnetothermal_in_room(node.get_label()):
24                         # Quindi non è utilizzabile e non lo inserisco nel grafo
25                         continue
26                     hmap_heuristic[node.get_label()] = self.underestimate_euclidean_distance(node.get_label(),
27                                         goals)
27                     for neighbor in node.get_neighbors():
28                         # Se il nodo è un ascensore
29                         if "ascensore" in neighbor:
30                             # Se il magnetotermico è attivo l'ascensore non funziona
31                             if da.check_magnetothermal_in_room(neighbor):
32                                 # Quindi non è utilizzabile e non lo inserisco nel grafo
33                                 continue
34                             # per ogni vicino del nodo analizzato aggiungo un arco
35                             distance = self.euclidean_distance(node.get_label(), neighbor)
36                             arcs.append(Arc(node.get_label(), neighbor, distance))
37
38             super().__init__(nodes=node_labels, arcs=arcs, start=start, goals=goals, hmap=hmap_heuristic)
39             print("Creazione del grafo di ricerca terminata con successo\n")
40

```

Listing 2: Costruzione del grafo di ricerca

calcola l'euristica verso gli obiettivi. Allo stesso tempo l'agente comunica con l'agente diagnostico per verificare che quando arriva ad esaminare un nodo ascensore questo sia in funzione, ovvero non sia scattato l'interruttore magnetotermico. Qualora questo fosse vero, il nodo viene "saltato" e ne lui ne i collegamenti ai suoi vicini vengono inseriti nel grafo. In questo modo, se un nodo non è disponibile, l'algoritmo di ricerca non lo considera proprio perché non viene inserito nel grafo. Per ogni nodo poi si esamina ogni vicino e si calcola la distanza dal nodo al vicino e quindi si aggiunge l'arco tra questi che avrà costo pari alla distanza calcolata. Allo stesso modo di prima l'agente comunica con l'agente diagnostico per verificare il funzionamento dell'ascensore, nel caso il vicino sia questo tipo di nodo.

Le funzioni utilizzate sfruttano tutte le interrogazioni sulla base di conoscenza.

## 4 Funzionamento

Tramite un'interfaccia testuale il sistema SAAC richiede all'utente cosa intende fare, offre due possibilità:

1. Cercare il percorso tra due posizioni dell'edificio sfruttando l'Agente ricerca cammino
2. oppure verificare lo stato dell'impianto elettrico tramite l'Agente diagnostico

```
-----
SAAC
-----
Menu:
Scegli una delle possibili operazioni (scrivi "q" per uscire):

1) Percorso più efficiente tra 2 punti
2) Stato della rete elettrica

Inserisci il numero dell'operazione da svolgere:
```

Se si sceglie la 1, il sistema per prima cosa chiamerà autonomamente l'agente diagnostico per verificare lo stato, quindi quest'ultimo stamperà eventuali anomalie. Dopo di che verrà richiesto di inserire il nodo di partenza e il nodo obiettivo, quindi verrà effettuata la ricerca e stampato a schermo il percorso più efficiente. Infine richiede se si vuole fare un'altra ricerca oppure tornare indietro.

```
Scelto la 1...
Riscontrate 25 anomalie nell'impianto elettrico.
Magnetotermico nella stanza 'ascensore_20_4_0' scattato
Magnetotermico nella stanza 'ascensore_20_4_1' scattato
-----
SAAC: Ricerca percorso più efficiente tra 2 punti
-----
Benvenuto :
Inserisci il luogo di partenza:
Partenza: corridoio_16_1_0
Inserisci il luogo di arrivo:
Arrivo: bagno_47_25_1
```

```
Magnetotermico nella stanza 'ascensore_20_4_0' scattato
Magnetotermico nella stanza 'ascensore_20_4_1' scattato
Creazione del grafo di ricerca terminata con successo

46 paths have been expanded and 35 paths remain in the frontier
Cammino trovato: corridoio_16_1_0 --> corridoio_11_4_0 --> corridoio_16_7_0 --> scala_20_10_0 --> scala_20_10_1 --> corridoio_24_7_1 --> corridoio_29_4_1 --> corridoio_29_12_1 --> corridoio_29_20_1 --> corridoio_33_17_1 --> corridoio_40_17_1 --> corridoio_48_17_1 --> corridoio_45_21_1 --> bagno_47_25_1
costo: 73.96499163953547

Vuoi cercare un percorso tra altri 2 nodi ? (y/n)
```

Se si sceglie la 2, invece, si viene posti davanti ad un'altra scelta:

1. Verificare lo stato dell'impianto
2. Effettuare una diagnosi su uno specifico dispositivo

```

Scelto la 2...
-----
SAAC: Stato dell'impianto elettrico
-----
Benvenuto :
Scegli una delle possibili operazioni (scrivi "q" per uscire):

1) Verifica lo stato dell'impianto elettrico
2) Diagnosi di un dispositivo

Inserisci il numero dell'operazione da svolgere:

```

Scegliendo la 1 l'agente esegue una diagnostica del sistema e mostra le anomalie trovate e se dei magnetotermici sono scattati, a questo punto chiede se si vuole vedere quali sono i dispositivi che non funzionano e se si vuole vedere un suggerimento per risolvere l'eventuale guasto. Infine torna indietro al menu precedente.

```

Scelto la 1...
Controllo dell'impianto...
Riscontrate 19 anomalie nell'impianto elettrico.
Magnetotermico nella stanza 'ascensore_20_4_0' scattato
Magnetotermico nella stanza 'ascensore_20_4_1' scattato
Vuoi vedere quali dispositivi non funzionano ? (y/n)
y
Il dispositivo 'heater_ascensore_20_4_0' è guasto. Probabile sovraccarico.
Il dispositivo 'light_corridoio_11_20_0' è guasto.
Il dispositivo 'light_corridoio_11_4_0' è guasto.
Il dispositivo 'light_corridoio_16_1_0' è guasto.
Il dispositivo 'light_corridoio_16_25_0' è guasto.
Il dispositivo 'light_corridoio_24_1_1' è guasto.
Il dispositivo 'light_corridoio_24_7_0' è guasto.
Il dispositivo 'light_corridoio_29_4_1' è guasto.
Il dispositivo 'light_corridoio_33_17_1' è guasto.
Il dispositivo 'light_corridoio_45_12_1' è guasto.
Il dispositivo 'light_corridoio_45_21_1' è guasto.
Il dispositivo 'light_corridoio_45_28_1' è guasto.
Il dispositivo 'light_corridoio_48_17_1' è guasto.
Il dispositivo 'light_scala_20_10_1' è guasto.
Il dispositivo 'light_scala_20_22_0' è guasto.
Il dispositivo 'light_scala_20_22_1' è guasto.
Il dispositivo 'light_ufficio_16_34_1' è guasto.
Il dispositivo 'light_ufficio_24_34_1' è guasto.
Il dispositivo 'light_ufficio_33_12_1' è guasto.
Vuoi vedere le possibili soluzioni per i dispositivi non funzionanti ? (y/n)

```

Scegliendo la 2 invece si può eseguire la diagnosi di uno specifico dispositivo inserito da tastiera, quindi il sistema mostrerà se il dispositivo è funzionante o meno, il problema rilevato ed il suggerimento per risolverlo:

```

Scelto la 2...
Inserire nome dispositivo di cui eseguire la diagnostica: heater_ascensore_20_4_0
Il problema con heater_ascensore_20_4_0 è overload.
Si è riscontrato un sovraccarico dovuto a heater_ascensore_20_4_0, la corrente è stata rimossa, scollegare dalla presa il dispositivo.
Scelto la 2...
Inserire nome dispositivo di cui eseguire la diagnostica: light_corridoio_11_4_0
Il problema con light_corridoio_11_4_0 è broken_light.
La soluzione per light_corridoio_11_4_0 è di sostituire la lampadina.

Il problema con light_area_relax_5_28_0 è replace_light.
La durata di vita di light_area_relax_5_28_0 sta per esaurirsi, è consigliata la sostituzione.
Premi invio per continuare...

```

## 5 Conclusioni

Il sistema è composto da due agenti: uno responsabile per la pianificazione dei percorsi tra le stanze e l'altro incaricato di monitorare lo stato dell'impianto elettrico dell'edificio. Questi agenti collaborano in modo sinergico, scambiando informazioni critiche per garantire un funzionamento ottimale dell'edificio.

I principali risultati ottenuti sono:

- **Efficienza nei Percorsi:** L'agente responsabile della pianificazione dei percorsi ha dimostrato un'elevata capacità nel trovare i cammini più efficienti tra le stanze dell'edificio. Questo è stato possibile grazie all'utilizzo di algoritmi di ricerca basati su conoscenza, che tengono conto di variabili come la disponibilità degli ascensori.
- **Monitoraggio dell'Impianto Elettrico:** L'agente diagnostico attraverso l'uso di regole basate su conoscenza, è in grado di rilevare anomalie e proporre soluzioni appropriate in tempo reale. In un ambiente reale questo contribuirebbe in modo significativo alla sicurezza dell'edificio e alla continuità del suo funzionamento.
- **Collaborazione Efficace:** Componente importante di questo sistema multi-agente risiede nella collaborazione sinergica tra gli agenti. L'agente di pianificazione dei percorsi interagisce costantemente con l'agente diagnostico per verificare la disponibilità delle risorse e prendere decisioni informate.
- **Adattabilità e Scalabilità:** Il sistema è stato progettato per essere altamente adattabile e scalabile. Può essere facilmente esteso per gestire edifici più grandi o complessi, e può essere aggiornato per incorporare nuove regole basate sulla conoscenza in risposta a cambiamenti nell'ambiente.

In sintesi, questo progetto dimostra come l'ingegneria della conoscenza possa essere applicata con successo per sviluppare sistemi intelligenti che migliorano l'efficienza e la sicurezza negli edifici, mostrando che con pochi accorgimenti ed integrazioni si possa migliorare di molto la "quality of life".

Future implementazioni potrebbero considerare ulteriori agenti per gestire altre funzionalità dell'edificio, come il controllo del clima o la sicurezza fisica, oltre che lo sviluppo di un'interfaccia grafica per rendere più piacevole l'interazione dell'utente.

## Riferimenti bibliografici

- [1] David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, 3 edition, 2023.