POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Scuola di Ingegneria Industriale e dell'Informazione

Proteggere obiettivi di valore supportati da allarmi spazialmente e funzionalmente imperfetti

Relatore: Prof. Nicola Gatti
Correlatore: Prof. Nicola Basilico
Correlatore: Ing. Giuseppe De Nittis

Tesi di Laurea di:
Matteo Daverio
Matricola 837674

Anno Accademico 2015-2016

Tutti sanno che una cosa è impossibile da realizzare,
finché arriva uno sprovveduto che non lo sa...
e la inventa.

Albert Einstein

Ringraziamenti

Durante il lavoro di ricerca e la stesura di questa tesi, molte persone mi sono state vicine e mi hanno supportato con il loro aiuto. Di tutto quello che avete fatto per me io vi ringrazio, senza di voi questo lavoro non sarebbe mai giunto al termine.

Il primo ringraziamento vorrei farlo a Nicola Gatti, a Nicola Basilico e a Giuseppe De Nittis. Grazie della vostra pazienza, del supporto datomi per scrivere questa tesi e di tutto il lavoro di ricerca svolto per arrivare a ottenere questi risultati. Senza di voi questo lavoro non sarebbe mai esistito. Grazie anche a tutti i professori che, nel bene e nel male, mi hanno fatto progredire nel mio percorso di studi fino a questo punto.

Un doveroso grazie va alla mia famiglia, a mamma, papà e Veronica, che per tutti questi anni mi hanno sopportato e supportato, sostenendomi nelle mie scelte e aiutandomi ad andare avanti. Grazie anche a nonno, nonna Ambrogina e nonna Silvia, per tutto quello che avete fatto per me e del tempo speso a ascoltare i miei lunghi ragionamenti mai troppo chiari su esami da dare e sul lavoro da svolgere per la tesi.

Arriviamo ora agli amici. Grazie Cipo, Johnny, Betty, Sara e Andre per la compagnia nei pomeriggi e nelle serate, tanto necessarie durante questo periodo universitario. Non sarebbe stata la stessa cosa senza di voi.

Ma soprattutto, voglio ringraziare i miei più cari e vecchi Amici, quelli con la A maiuscola, che non sono mai mancati per me e che farebbero qualsiasi cosa per aiutarmi.

Grazie Fede, per tutti i discorsi seri, i dibattiti videoludici, gli estati passati insieme quando tutti gli altri lavoravano e tutti i progetti iniziati con grande convinzione e mai conclusi; magari prima o poi riusciremo anche a portarne a termine qualcuno.

Grazie N, perchè sei sempre presente in ogni cosa che facciamo. Perchè non potrei avere un compagno di forchetta e un support migliore con cui passare il mio tempo.

Grazie John, per le serate horror e gli anime visti insieme. Per la pazzia che metti in ogni cosa che fai, per gli gnomi armati di ascia e i galli cedroni.

Grazie Tias, per tutte le risate che mi fai fare, per essere sempre disponibile a tenere compagnia a chiunque e per offrirci sempre un cosi fresco punto di ritrovo.

Ringrazio anche tutti i compagni universitari, con cui tanto ho condiviso in questo travagliato cammino. Un grazie particolare va a Cimbo, Cerez, Andre, Leo, Vale e Mirko per i bei momenti passati insieme.

Sommario

La Teoria dei Giochi sta assumendo sempre più rilevanza nell'ambito dell'intelligenza artificiale, in quanto consente di modellare in modo verosimile la realtà. Una importante sottocategoria di tale teoria consiste nei *Patrolling Security Games*.

Con questo nome identifichiamo tutti quei giochi basati sulla protezione di determinati obiettivi. Si ha infatti un'area modellata tramite un grafo, in cui vi sono degli obiettivi sensibili. A protezione di essi avremo un difensore, che può essere una pattuglia umana oppure un'unità robotica. Essi devono proteggere i punti di maggior interesse da un attaccante, ovvero un malintenzionato che desidera sottrarre o danneggiare tali obiettivi.

Nel caso da noi preso in esame, assumiamo che il malvivente possa osservare ogni mossa del difensore prima di decidere come agire. Dall'altro lato, la guardia sarà supportata da un sistema di allarmi. Per rendere il tutto verosimile, i sensori saranno spazialmente e funzionalmente imperfetti. Con ciò intendiamo che ogni allarme sarà assegnato a più obiettivi contemporaneamente e quindi non darà un'informazione precisa sul luogo in cui il criminale sta commettendo il reato; inoltre avremo che i sensori possono fallire la rilevazione con una probabilità α, non segnalando l'attacco da parte del malintenzionato.

Supporteremo tali analisi con delle valutazioni sperimentali, provando il modello con casi differenti che possano mostrare la sua scalabilità e come si modificano le performance in relazione ai vari parametri del problema.

Abstract

Game Theory is obtaining more and more importance in the field of artificial intelligence, because it allows to model reality in a realistic way. An important subcategoryof this theory consists in *Patrolling Security Games*.

With this name we identify all those games based on the protection of certain objectives. In fact we have an area modeled by a graph, in which there are some sensible targets. To protect them we will have a defender, which can be a human or a robotic patrol unit. They must protect the points of major interest for an attacker, who wants to steal or damage these objectives.

In the case we examined, we assume that the criminal can observe every moves of the defender before deciding how to act. On the other hand, the guard will be supported by an alarm system. To make this plausible, the sensors will be spatially and functionally imperfect. With that definition we mean that each alerm will be assigned to multiple targets simultaneously and therefore will not give a precise information about the place in which the criminal is committing the attack; we will also have that sensors can fail detection with a probability α, where signals don't report the attack by the attacker.

We will support all the analysis with experimental evaluations, testing the model with different cases that can show its scalability and how the performance will change in relation to the various parameters of the problem.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

"I computer sono inutili.
Ti sanno dare solo risposte."

Pablo Picasso

1.1 Dove tutto ha inizio

Ipotizziamo di essere gli addetti alla sorveglianza di un importante museo. Per sorvegliare, ipotizziamo di possedere un servizio di allarmi che possa rilevare dei furti posizionato per coprire più opere di valore contemporaneamente. Avremo inoltre un'unità di pattuglia che si occuperà di muoversi all'interno del museo per controllare che tutto sia al sicuro.

Se un ladro dovesse riuscire ad infiltrarsi nell'edificio, seguendo la guardia per conoscerne il percorso di pattuglia, potrà poi studiare una strategia che possa consentirgli di portare a termine il furto senza rischiare la cattura. Tuttavia, anche la guardia sa che, da un momento all'altro, chiunque può entrare nel museo e trafugare un'opera.

Il malintenzionato ha quindi come scopo quello di trafugare l'opera, osservando e studiando l'ambiente per poter determinare tutte le informazioni che potrebbe utilizzare a suo vantaggio. Dall'altro lato, la guardia vuole proteggere le opere, impedendone il furto. Per raggiungere questo scopo può sfruttare varie risorse che possano agevolarlo, come allarmi o telecamere. Non può prevedere però quando e dove l'attacco avrà luogo, può solo scegliere come muoversi per cercare di ottimizzare le proprie reazioni a qualsiasi tipo di variazione nell'ambiente circostante.

Il problema di decidere quali sono le migliori azioni della pattuglia rientra nella categoria dei *Patrolling Security Games*, i quali fanno parte della più estesa Teoria dei Giochi [6].

In breve, i giochi di sicurezza trattano in generale delle situazioni sopra descritte. Avremo infatti un ladro, che riveste il ruolo di attaccante e una guardia, che invece avrà il compito di gestire la difesa. Si verificherà quindi che il malintenzionato avrà come scopo quello di derubare uno dei vari obiettivi, mentre il difensore vorrà evitare che ciò accada.

1.2 Cosa vogliamo ottenere

Parte di questo lavoro già esiste ed è stato studiato in precedenza, noi vogliamo estenderlo per gestire un maggior numero di casistiche. Principalmente, vorremmo poter studiare anche casi in cui viene introdotto un sistema di allarme, il quale genera segnali che possono essere affetti con una certa probabilità da falsi negativi, ovvero mancanza di rilevazione durante un tentativo di furto.

Questa estensione è stata decisa per rendere il modello più realistico, infatti può capitare che un segnale, con una probabilità α, non effettui una corretta rilevazione. Sempre con l'ipotesi di analizzare casi realistici i segnali, poichè costosi, verranno utilizzati per sorvegliare più obiettivi contemporaneamente. Questo inserirà un'incertezza spaziale agli allarmi. Ciò comporterà che se un segnale dovesse attivarsi, non potremo sapere a quale obiettivo esso si riferisce, ma ci diverrà noto solo l'insieme di obiettivi protetti da tale segnale.

Questa ricerca ha quindi come obiettivo quello di valutare le possibili strategie dell'unità utilizzata per il pattugliamento. Vedremo come queste strategie variano a seconda delle modifiche all'ambiente da proteggere, ad esempio inserendo più obiettivi, aumentando i collegamenti tra loro, variando il numero di segnali oppure modificando le probabilità di falso negativo.

1.3 Struttura della tesi

Per questa tesi è stata utilizzata la seguente struttura:

- il Capitolo 2 si occupa di mostrare lo stato dell'arte, descrivendo principalmente i concetti principali legati alla teoria dei giochi, focalizzandosi su modelli e metodi risolutivi utilizzati in questa ricerca. Verrà

posta enfasi anche su tutto ciò che già è stato fatto nell'ambito dei *Security Games*;

- il Capitolo 3 descrive il problema che vogliamo affrontare in questa tesi. Espone in maniera formale cosa desideriamo ottenere e analizza più nello specifico i giochi che necessitano di essere risolti per poter conoscere le strategie ottime per il difensore;
- il Capitolo 4 analizza gli algoritmi studiati per risolvere i problemi esposti nel capitolo precedente. Qui verranno proposte le principali alternative su come ottenere le strategie che possano indicare come il difensore debba comportarsi per migliorare la propria utilità;
- il Capitolo 5 si occupa della fase di sperimentazione. In esso vengono infatti descritte le istanze utilizzate per gli esperimenti. Essi sono svolti in modo tale da valutare il modello presentato e gli algoritmi studiati per risolverlo.
- il Capitolo 6 espone le conclusioni tratte su questo lavoro di tesi, mostrando le possibili future applicazioni per estendere tale ricerca.

Capitolo 2

Stato dell'arte

"Se l'uomo non sapesse di Matematica
non si eleverebbe di un sol palmo da terra."

Galileo Galilei

Per comprendere appieno gli argomenti trattati in questa ricerca, verranno esposti in questo capitolo tutti i concetti fondamentali. Ci si soffermerà in particolare sulla trattazione di argomenti legati alla teoria dei giochi e in particolare ai *Patrolling Security Games*.

2.1 Teoria dei Giochi

La teoria dei giochi ha origini non troppo antiche. Nasce infatti agli inizi del 900 con due testi di John Von Neumann, con cui getta le basi [16] e poi si addentra ed esplora questo nuovo ambito [17]. Ai giorni nostri, sempre più spesso la teoria dei giochi viene utilizzata negli ambiti più disparati, dal campo economico e finanziario a quello strategico-militare, dalla politica alla sociologia, dalla psicologia all'informatica, dalla biologia allo sport. Non è dunque difficile capire l'importanza di questa disciplina che, al giorno d'oggi, è usata per studiare e analizzare le decisioni individuali di un soggetto in situazioni di interazione strategica con altri agenti razionali [12]. In questa sezione partiremo identificando il significato di gioco, seguendo con vari modelli utilizzati in questa ricerca.

2.1.1 In cosa consiste un gioco?

Definizione 1. *Un gioco è un modello di interazione caratterizzato da:*

- $N = 1, 2, \ldots, n$, *ovvero l'insieme di tutti i giocatori, dove 1, 2, ..., n sono i giocatori.*
- $A = \{A_1, A_2, \ldots, A_n\}$, *ovvero l'insieme di tutte le azioni che possono essere compiute dai vari agenti. Più precisamente, l'insieme* A_i *rappresenta le azioni che l'agente i può compiere.*
- X, *l'insieme dei possibili esiti del gioco.*
- $f : A_1 \times A_2 \times \cdots \times A_n \to X$, *la funzione degli esiti, ovvero la funzione che, data una particolare sequenza o insieme di azioni eseguite dai giocatori, ne associa un esito corrispondente.*
- U_i, *ovvero l'utilità del gioco per il giocatore i. Indica quanto al giocatore possa essere conveniente partecipare al gioco. In questa tesi verrà spesso utilizzato come sinonimo di payoff e indicato con il simbolo* π.
- $u_i : X \to \mathbb{R}$, *ovvero la funzione di utilità, che associa ad ogni possibile esito del gioco un valore* $v \in \mathbb{R}$ *per ciascun giocatore.*

L'esito di un gioco si caratterizza, per ogni giocatore, dall'insieme di azioni che lo porterebbero a massimizzare la propria utilità. Per chiarire il comportamento che i vari giocatori assumono, analizziamo un noto esempio [8], come formalizzato da Albert W. Tucker [15].

Esempio 1. *Due persone sono sospettate per aver compiuto un crimine. Il giudice avanza una proposta ai due prigionieri, i quali non possono comunicare tra loro. Dà loro la possibilità di confessare i crimini dell'altro, oppure non confessare. Le possibili situazioni sono le seguenti:*

- *Se nessuno dei due confessa, verranno entrambi arrestati per un reato inferiore, con una pena pari a 1 anno di prigione.*
- *Se entrambi confessano, verranno arrestati per una durata ridotta a 5 anni, come premio per l'onestà.*
- *Se solo uno confessa, allora chi ha confessato sarà libero e l'accusato avrà una pena di 7 anni.*

Appare chiaro che l'esito più favorevole ai giocatori è dato dalla non confessione di entrambi, tuttavia il risultato sarà più verosimilmente che tutti e due confesseranno, scontando una pena di 5 anni entrambi.

Questo esempio è utile a capire che i giocatori compieranno le loro azioni in modo *egoistico* e *razionale*.

- I giocatori si comportano in maniera egoistica, cioè cercheranno sempre di compiere l'azione che massimizza la loro utilità, senza prendere in considerazione la soddisfazione degli altri giocatori.

- I giocatori si comportano in maniera razionale, cioè se conoscono i meccanismi che regolano il gioco li applicheranno per conoscere come gli altri giocatori si comporteranno e di conseguenza scegliere le proprie azioni in maniera più efficiente [3]. Questo concetto si applica di conseguenza a tutti i giocatori.

In questa tesi verranno prevalentemente analizzati giochi *non-cooperativi* [10], ovvero giochi in cui si studiano le interazioni tra agenti singoli, *ad informazione perfetta* [17], ovvero giochi nel quale tutto è osservabile, e *deterministici*, ovvero giochi nei quali ogni sequenza di azioni porta con certezza ad un certo esito.

2.1.2 Giochi in forma estesa

Un gioco in forma estesa [7] è caratterizzato dalla formalizzazione matematica di tutte le informazioni rilevanti del gioco stesso. Più precisamente è dato da:

- Lo stato iniziale del gioco

- Tutte le possibili evoluzioni, ovvero tutte le mosse consentite in ogni possibile situazione

- Tutti i possibili esiti del gioco e le preferenze di ogni giocatore verso ognuno di essi.

Solitamente queste informazioni sono raccolte in grafi ad albero, come quello in Figura 2.1, che mostra la classica rappresentazione di albero di gioco.

- I nodi corrispondono a stati del gioco. Essi, nell'esempio, sono indicati con una coppia $i.j$, dove i indica che il giocatore che deve compiere la successiva azione, mentre j indica il nodo.

- Gli archi indicano le possibili azioni che i giocatori possono compiere in un determinato stato.

- Le foglie rappresentano l'utilità dei giocatori dato quel particolare esito. Più precisamente u_{ij} rappresenta l'utilità del giocatore j quando si verifica l'esito i.

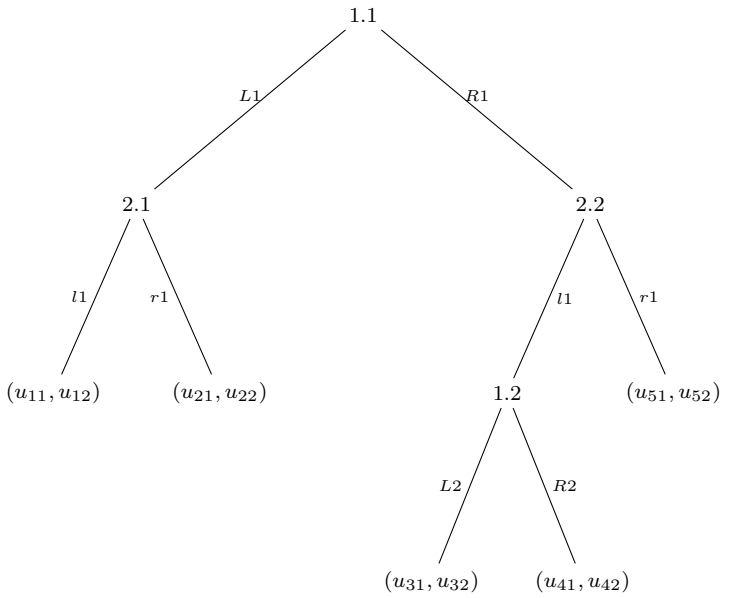


Figura 2.1: Esempio di rappresentazione ad albero per i giochi in forma estesa.

Per risolvere questo tipo di giochi si usa una tecnica chiamata *backward induction*, o *induzione a ritroso*. Questa tecnica consiste nel procedere a ritroso dalle foglie alla radice, selezionando di volta in volta l'azione ottimale per ciascun giocatore. Per chiarire meglio questo concetto, mostriamo ora un semplice esempio, inserendo dei valori numerici alle foglie dell'albero sopra rappresentato.

Esempio 2. *L'albero con le relative utilità è rappresentato in Figura 2.2.*

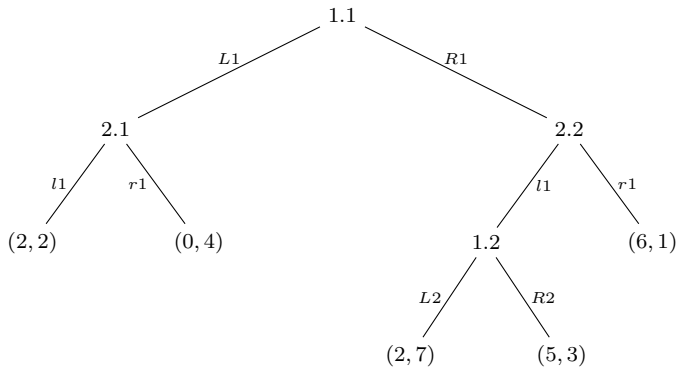


Figura 2.2: Rappresentazione ad albero per i giochi in forma estesa con utilità.

Dalla Figura 2.2 possiamo facilmente vedere che applicando la backward induction, al nodo 1.2 *il giocatore 1 esegue la mossa* $R2$*, che conferisce un'utilità di 5, contro la mossa* $L2$*, che gli fornisce invece un'utilità di 2. Possiamo quindi semplificare l'albero, riportando gli esiti di cui siamo certi, come mostrato in Figura 2.3.*

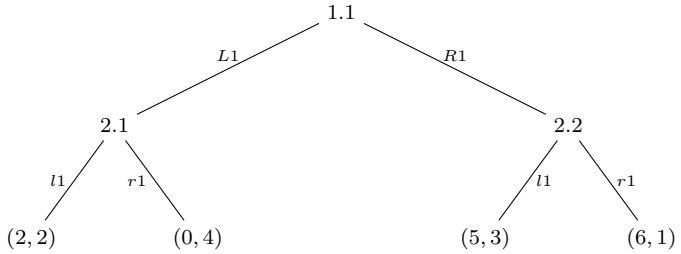


Figura 2.3: Albero semplificato per il gioco in forma estesa.

A questo punto vediamo che:

- *Al nodo* 2.1 *il giocatore 2 esegue la mossa* $r1$ *che gli conferisce un'utilità di 4, superiore all'utilità di 2 data dalla mossa* $l1$*;*
- *Al nodo* 2.2 *il giocatore 2 esegue la mossa* $l1$ *in quanto fornisce un'utilità di 3, rispetto alla mossa* $r1$ *che fornisce un'utilità di 1.*

L'albero risultante è dunque rappresentato in Figura 2.4.

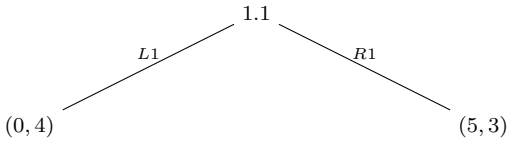


Figura 2.4: Albero completamente semplificato per il gioco in forma estesa.

A questo punto è chiaro che il giocatore 1 esegue la mossa $R1$*, che gli fornisce un'utilità di 5. L'esito del gioco è dunque* $(5, 3)$*, ovvero il giocatore 1 avrà un'utilità di 5 e il giocatore 2 avrà un'utilità di 3.*

Come abbiamo potuto vedere dall'Esempio 2 abbiamo a disposizione uno strumento potente per la risoluzione di questo tipo di problemi; tuttavia esistono anche giochi che non possono essere risolti tramite questo metodo. Diventa perciò necessario l'utilizzo di altri metodi.

2.1.3 Giochi in forma strategica

Al contrario della forma estesa, più adatta per giochi a turni o comunque in cui esiste una sequenzialità, la forma strategica [17] è più adatta per quei giochi caratterizzati da un turno solo, in cui spesso i giocatori agiscono simultaneamente. Per questi problemi si ricorre a una rappresentazione matriciale, inserendo nelle celle l'utilità di ciascun giocatore nel caso in cui si dovesse verificare quel determinato esito del gioco. Per chiarire questo concetto, facciamo riferimento all'Esempio 1

Esempio 3. *Chiamiamo i due prigionieri A e B rispettivamente. Indichiamo inoltre con C e NC le possibili azioni di Confessare e Non Confessare. La matrice del gioco è rappresentata in Tabella 2.1.*

		B	
		C	NC
A	C	$(5,5)$	$(0,7)$
	NC	$(7,0)$	$(1,1)$

Tabella 2.1: Rappresentazione matriciale del Dilemma del Prigioniero.

Appare intuitivo anche qui capire che l'esito del gioco porterà entrambi a confessare.

A questo punto diventa necessario capire come i giocatori devono scegliere le proprie mosse per massimizzare la propria utilità.

Definizione 2. *Una strategia, indicata da un vettore $\underline{\sigma_i}$, è la probabilità che il giocatore i giocherà le proprie azioni. Per questo vettore deve valere che:*

- $\forall a \in A_i \quad \sigma_{i,a} \geq 0$,
- $\sum_{\forall a \in A_i} \sigma_{i,a} = 1$.

Una strategia è detta *pura* se esiste un elemento pari a 1 e tutti gli altri sono pari a 0. Se cosi non fosse, la strategia è detta *mista*. Una strategia è detta *completamente mista* se ogni elemento di σ_i è strettamente positivo.

2.1.4 Metodi risolutivi

Per risolvere questa categoria di giochi si possono utilizzare varie tecniche, alcune più generiche, altre più specifiche per determinate sottoclassi di problemi. I metodi principali verranno descritti in questa sezione.

Eliminazione di strategie dominate

Vediamo ora il concetto di strategie dominate [5].

Definizione 3. *Sia i un giocatore e $a \in A$ un'azione che il giocatore i può compiere. Allora a è strettamente dominata se e solo se esiste una strategia x_i tale che:*

$$\forall x_{-i}, \qquad \mathbb{E}[U_i(x_i, x_{-i})] > \mathbb{E}[U_i(a_i, x_{-i})]$$

Significa quindi che se una strategia è sempre migliore delle altre indipendentemente dalle scelte degli altri agenti, allora quella è la strategia ottimale. Per ogni giocatore si analizzeranno quindi le sue azioni, e si scarteranno se porteranno sempre, qualsiasi sia la scelta degli avversari, a un'utilità minore di quella data da altre azioni. Vediamo anche questo concetto applicato ad un esempio.

Esempio 4. *Prendiamo ancora in analisi il Dilemma del Prigioniero, esposto nell'Esempio 1. Modifichiamo però le utilità, rendendole negative, in modo tale che i giocatori massimizzando il proprio guadagno otterranno il minor periodo in prigione possibile. Riportiamo quindi la matrice delle utilità dei due giocatori, in Tabella 2.2.*

		B	
		C	NC
A	C	$(-5, -5)$	$(0, -7)$
	NC	$(-7, 0)$	$(-1, -1)$

Tabella 2.2: Rappresentazione matriciale del Dilemma del Prigioniero.

Iniziamo la nostra analisi di questo gioco dal primo giocatore. Egli infatti conosce la matrice del gioco, con le utilità di entrambi. A lui sono disponibili solo due mosse, Confessare o Non Confessare, che saranno le stesse mosse selezionabili anche dall'altro prigioniero. Analizzando le utilità possibili, se il giocatore decidesse di confessare, otterrebbe -5(ovvero 5 anni di prigione) se l'avversario dovesse compiere la stessa mossa, mentre otterrebbe 0 nel caso in cui il secondo giocatore non dovesse confessare. Nel caso in cui, invece, decidesse di compiere l'azione di non confessare, i risultati sarebbero -7(ovvero 7 anni di prigione) nel caso in cui l'altro prigioniero confessasse, oppure -1(quindi 1 anno di prigione) nel caso di non confessione anche

dalla controparte. Come già discusso, la soluzione migliore per entrambi è data dalla scelta di non confessare per ambo le parti, tuttavia i giocatori non possono accordarsi, e le utilità del primo giocatore, come appena analizzato, sono -5 *o* 0 *nel caso di confessione,* -7 *o* -1 *nel caso di non confessione. E' chiaro che in questo caso, qualsiasi sia la mossa compiuta dal secondo giocatore, le utilità che il primo giocatore otterrebbe lo porterebbero a compiere sempre la mossa di confessare. Si può pertanto, per semplificazione, eliminare dalla matrice del gioco la scelta di non confessare per il primo giocatore, in quanto non verrà mai scelta come azione. La nuova matrice del gioco è mostrata in Tabella 2.3.*

		B	
		C	NC
A	C	$(-5, -5)$	$(0, -7)$

Tabella 2.3: Rappresentazione matriciale semplificata del Dilemma del Prigioniero.

In questo caso, anche per il secondo giocatore si pone la stessa scelta, ovvero Confessare o Non Confessare. Ribadiamo anche il concetto che in quanto giocatori razionali essi conoscono le utilità del gioco e sono in grado entrambi di arrivare alla medesima conclusione su quale sia la mossa migliore da compiere. Dunque, il secondo giocatore conosce che il primo deciderà di confessare, quindi gli si presentano due scelte. La prima è quella di confessare, con un'utilità di -5*, la seconda è quella di non confessare, con un'utilità di* -7*. E' chiaro che la scelta migliore è quindi di confessare, come rappresentato in Tabella 2.4.*

		B
		C
A	C	$(-5, -5)$

Tabella 2.4: Rappresentazione matriciale completamente semplificata del Dilemma del Prigioniero.

La soluzione del gioco è dunque che entrambi i prigionieri confessino, scontando una pena di 5 *anni a testa.*

Con questo esempio abbiamo potuto capire quanto l'eliminazione di strategie dominate sia uno strumento potente per la risoluzione di giochi. E' possibile tuttavia che l'eliminazione iterata di strategie, ovvero la rimozione di possibili mosse da parte di un giocatore su una tabella già semplificata, porti a esiti finali differenti. Infatti, modificando l'ordine di eliminazione può capitare che alcune strategie non vengono dominate, modificando quindi l'esito. Questo metodo ha però una grande limitazione, ovvero non può essere applicato in tutti i casi, in quanto spesso i giochi non possiedono strategie dominate e quindi bisogna necessariamente ricorrere ad altre tecniche. Vediamone qualcuna.

Equilibri di Nash

Definizione 4. *Un gioco non cooperativo a due giocatori in forma strategica è definito dalla quadrupletta* $(X, Y, f : X \times Y \to \mathbb{R}, g : X \times Y \to \mathbb{R})$. *Un equilibrio di Nash per il gioco è una coppia* $(\overline{x}, \overline{y}) \in X \times Y$ *tale per cui:*

- $f(\overline{x}, \overline{y}) \geq f(x, \overline{y})$ *for all* $x \in X$*;*
- $g(\overline{x}, \overline{y}) \geq g(\overline{x}, y)$ *for all* $y \in Y$*.*

Ciò che possiamo intuire è che quindi un equilibrio di Nash è un profilo di strategie dove nessun giocatore ha incentivo a deviare unilateralmente.

Questa definizione indica che, per ogni possibile mossa del secondo giocatore, il primo calcolerà la miglior azione compibile da parte sua. Questo processo viene svolto da entrambi, e le coppie che saranno la scelta migliore per entrambi i giocatori vengono identificate come equilibri. Gli equilibri di Nash sono tutte le coppie (x, y) tali per cui, assumendo che l'altro giocatore compirà l'azione che ha annunciato, non ha incentivo a modificare la propria mossa; dall'altra parte, anche l'altro giocatore non avrà incentivo a deviare dall'esito che si otterrebbe, per la stessa ragione.

Il concetto di equilibrio di Nash è fondamentale nella Teoria dei Giochi, specialmente se rafforzato con il seguente teorema.

Teorema 1. *Ogni gioco finito, ovvero con un numero finito di giocatori e strategie, ammette almeno un equilibrio di Nash in strategie miste.*

MinMax

Un altro metodo risolutivo molto potente è noto come *minmax*. Esiste una particolare categoria di giochi, detti giochi a somma zero [17], o a somma

costante, nei quali minmax ha un'importante peculiarità, ovvero coincide con l'equilibrio di Nash.

Definizione 5. *Un gioco si dice a somma zero quando, chiamato W l'insieme degli esiti, $u_i(w)$ l'utilità del giocatore i per l'esito w, N l'insieme dei giocatori, si ha*

$$\sum_{i \in N} u_i(w) = 0 \quad \forall w \in W$$

Diventa intuitivo osservare che nel caso di due giocatori i, j, l'utilità di uno sarà l'opposto dell'altra, quindi $u_i = -u_j$. Pertanto, non sarà necessario riportare entrambi i valori, in quanto riportando la sola utilità del giocatore i si ricaverà facilmente quella del giocatore j.

Da questa intuizione si ricava anche che mentre il giocatore i cercherà di massimizzare il valore, l'agente j, poichè la sua utilità è l'opposto di quella del primo soggetto, deve invece fare in modo che quest'ultimo guadagni la minore utilità possibile. E' quindi evidente che il primo agente, per effetto delle scelte del secondo, guadagnerà il massimo dei minimi sulle righe delle matrice, mentre il secondo, per effetto del primo, guadagnerà il minimo dei massimi sulle colonne.

Definizione 6. *Una coppia di strategie $(\bar{x}, \bar{y})$ è una soluzione minmax per un gioco a somma zero se*

$$f(x, \bar{y}) \leq f(\bar{x}, \bar{y}) \leq f(\bar{x}, y) \quad \forall x \in X, y \in Y$$

Chiariamo questo concetto con un esempio, tratto da [8].

Esempio 5. *Prendiamo ad esempio questo semplice gioco a somma zero, rappresentato in Tabella 2.5.*

	A	B	C
A	2	3	4
B	0	7	1
C	1	0	5

Tabella 2.5: Matrice di un gioco a somma zero.

Come già discusso, il primo giocatore otterrà il valore massimo tra i valori minimi di ogni riga. Ragioniamo sull'esempio. Se il primo giocatore scegliesse l'azione B, nella speranza di ottenere l'utilità di 7, il secondo giocherà invece A, che è la mossa più vantaggiosa per lui. Allo stesso modo

se giocasse A, il secondo giocatore giocherebbe A e nel caso in cui il primo scegliesse C, il secondo opterebbe per B. I possibili risultati per il giocatore 1 sono quindi $2, 0, 0$*. E' pertanto immediato capire che non ha convenienza a giocare B o C, rendendo di fatto A la mossa migliore per lui. Ragionando allo stesso modo procediamo per il secondo giocatore, ricordando che però il suo scopo è minimizzare il valore e non massimizzarlo. Le possibili utilità ottenibili sono* $2, 7, 5$*. Anche in questo caso, siccome vogliamo minimizzare il valore, la scelta migliore ricade sulla mossa A, portando quindi il gioco ad avere come scelta migliore la mossa A per entrambi i giocatori, con un'utilità di* 2 *per il primo giocatore e* -2 *per il secondo.*

2.1.5 Leader-Follower

Esiste un particolare concetto di soluzione, detto *Leader-Follower* o giochi di Stackelberg [18], che sono caratterizzati dalla particolarità che prima di giocare un giocatore dichiara la propria strategia. Infatti vi è un primo giocatore, detto *leader*, che dichiara la propria strategia e uno o più giocatori, detti *follower*, che preso atto della strategia dichiarata, ne sceglieranno una propria. Come già discusso, noi ipotizziamo i giocatori egoisti e razionali, pertanto il secondo agente effettua dei calcoli che gli consentono di sapere qual è la miglior azione in relazione alla mossa del primo soggetto. Questi calcoli però possono essere fatti anche dal leader, il quale può di conseguenza sfruttare questa conoscenza per massimizzare il proprio guadagno, influenzando il comportamento del follower con la propria mossa. Utilizziamo un esempio per chiarire il concetto, sempre tratto da [8].

Esempio 6. *Due aziende producono lo stesso prodotto in quantità* q_1 *e* q_2 *rispettivamente, in un mercato che richiede una quantità massima a di tale prodotto. Dato c come costo unitario di produzione e* $a - (q_1 + q_2)$ *il prezzo unitario del prodotto, possiamo assumere:*

- *Se il leader annuncia la quantità* q_1*, il follower cercherà di massimizzare la propria utilità* $q_2(a - (q_1 + q_2)) - cq_2$*.*
- *La quantità ottimale per il follower risulta quindi* $\overline{q_2}(q_1) = \frac{a-q_1-c}{2}$*.*
- *Il leader, conoscendo la quantità ottimale del follower, massimizzerà la propria utilità, ottenendo quindi* $\overline{q_1} = \frac{a-c}{2}$*.*
- *Sostituendo quindi tutti i valori scopriamo che la quantità ottimale per il follower sarà* $\overline{q_2} = \frac{a-c}{4}$*. Le utilità per entrambi i giocatori saranno* $u_1 = \frac{(a-c)^2}{8}$ *e* $u_2 = \frac{(a-c)^2}{16}$ *rispettivamente.*

Appare chiaro che se un giocatore può intraprendere una strategia che lo porterà in posizione di leader, egli compierà tale scelta, in quanto la sua utilità risulterebbe sicuramente superiore a quella che avrebbe ottenuto in posizione di follower.

2.2 Patrolling Security Games

La sottocategoria di giochi su cui abbiamo modellato il problema e che quindi più ci interessa analizzare per questa tesi è detta dei Patrolling Security Games(PSG)[1].

Definizione 7. *Un PSG è composto dai seguenti elementi:*

- *Due giocatori, un attaccante $\mathcal{A}$ e un difensore $\mathcal{D}$;*
- *Un grafo connesso e non orientato $G = (V, E)$, in cui ogni arco $(i, j) \in E$ richiede un turno per essere attraversato;*
- *Un sottoinsieme dei nodi del grafo, detto insieme dei Target $T \subseteq V$, che sono rilevanti per $\mathcal{A}$ e $\mathcal{D}$;*
- *Un valore $\pi(t) \in (0, 1]$, che rappresenta il valore del singolo target;*
- *Un tempo di penetrazione $d(t) \in \mathbb{N}$, che indica il numero di turni necessari per completare con successo un attacco al target t.*

Per chiarire questo concetto, proponiamo un esempio.

Esempio 7. *Ipotizziamo di avere un luogo da proteggere, ad esempio un quartiere in cui sono presenti un certo numero di banche o altri edifici per attività finanziarie, nei quali sono contenuti quindi degli obiettivi di valore. Possiamo facilmente identificare come difensore una pattuglia che si occuperà di controllare queste costruzioni e come attaccante un malintenzionato deciso a infiltrarsi in un edificio e sottrarre i beni di valore. Come grafo possiamo mappare il quartiere, usando come vertici gli incroci e come archi le strade. Di questo grafo possiamo identificare come target i vertici in prossimità degli edifici sensibili. A questi assegnamo poi un valore $\pi(t)$ dato dall'importanza dei beni che esso contiene, normalizzato per ottenere un valore tra 0 e 1. Assegnamo inoltre un tempo di penetrazione $d(t)$, dato dal tempo necessario al ladro per infiltrarsi, sottrarre i beni e fuggire. Chiaramente il difensore deve riuscire a impedire al ladro di completare il suo attacco, perciò dovrebbe trovare il modo per passare per i target almeno una volta ogni $d(t)$ unità di tempo, altrimenti l'attaccante potrebbe facilmente attendere il passaggio della pattuglia per poi attaccare indisturbato.*

Per proseguire nell'analisi di questo gioco, dobbiamo definire l'utilità per il difensore e per l'attaccante. Come già detto, il ladro cercherà di ottenere il bottino di valore maggiore per poi scappare, mentre il difensore cercherà di proteggere tutti gli obiettivi, in modo tale da impedire al malintenzionato di attaccare, oppure per coglierlo sul fatto. Questo modello è quindi caratterizzato da utilità contrapposte, caratterizzandolo come gioco a somma costante[1]. Più precisamente, per l'attaccante avremo:

- 0: se l'attaccante viene catturato o non riesce a compiere l'attacco;
- $\pi(t)$ se l'attaccante riesce a portare a termine con successo l'attacco al target t.

Di contro, per il difensore avremo:

- 1: se il difensore riesce a proteggere tutti i target, in modo tale che non vengano attaccati, oppure se cattura l'attaccante;
- $1 - \pi(t)$: se l'attaccante completa con successo l'attacco al target t, ovvero se il difensore non passa per il target t per più di $d(t)$ unità di tempo.

Analizziamo ora le mosse consentite ai due giocatori, per chiarire cosa essi possono fare a ogni turno del gioco. L'attaccante può:

- Attendere un turno;
- Attaccare il target t.

Il difensore invece:

- Restare fermo nel vertice v;
- Spostarsi dal vertice v al vertice v' tramite l'arco e che li collega.

Chiaramente questo gioco può durare per un lungo periodo di tempo, rendendo complesso calcolarne la soluzione. Per semplificare la risoluzione si assumerà che la guardia percorrerà un ciclo di pattugliamento deterministico, ovvero una sequenza di vertici $(v_1, v_2, ..., v_n) \in V$ seguiti ripetutamente dal difensore per tutta la durata del gioco, come descritto da [1].

[1] Più precisamente si tratta di un gioco *Leader-Follower*, ma come spiegato in [1] esso si può ricondurre in un gioco a somma costante.

Introduzione di allarmi

Per rendere più realistico il modello, utilizziamo un'interessante estensione del problema [11], aggiungendo un sistema di allarmi spazialmente imperfetti. Ipotizziamo inoltre che ogni allarme, se rileva un malintenzionato intento a rubare o danneggiare un obiettivo, invia un segnale alla guardia in modo da portarlo a conoscenza del fatto. Per allarmi spazialmente imperfetti intendiamo che ogni segnale è utilizzato per coprire una determinata area più o meno estesa. Esso ci darà informazione che qualcosa di anomalo si sta verificando nell'area a cui è dedicato, senza però darci informazioni precise circa il punto in cui si sta verificando l'infiltrazione. E' perciò compito del difensore focalizzarsi sul perlustrare tale area alla ricerca dell'attaccante. In questo modo possiamo utilizzare un numero limitato di allarmi, che saranno comunque in grado di scoprire eventuali attacchi.

Nel nostro modello del gioco, per gestire i segnali, introduciamo:

- $S = s_1, s_2, ..., s_n$, l'insieme dei segnali che il sistema di allarme è in grado di generare;
- $p(s|t)$, la probabilità che, quando il target t è sotto attacco, venga attivato il segnale s. Questa probabilità deve essere definita per ogni target t e per ogni segnale s.

2.2.1 Lavori significativi

Recentemente, varie applicazioni legate alla risoluzione di problemi di sicurezza in ambienti reali sono state proposte. I casi più significativi e degni di nota sono legate alla distribuzione di posti di blocco della polizia nell'aeroporto internazionale di Los Angeles [13], oppure riguardo alla distribuzione dello spazio aereo americano [14].

Altri lavori sono stati svolti su vari ambiti. Nel caso di più unità per la difesa, ricerche sono state svolte per tenere in conto i problemi di comunicazione per migliorare il coordinamento tra le varie risorse [2].

Sono stati anche effettuati studi per trattare lo scenario in cui l'attaccante ha la possibilità di eseguire attacchi multipli [4]. Allo stesso modo, sono state trattate le casistiche in cui più difensori pattugliano l'area [9].

Capitolo 3

Il problema

"In matematica non si capiscono le cose.
Semplicemente ci si abitua ad esse."

John Von Neumann

In questa sezione tratteremo il modello del problema che andremo ad affrontare. Più precisamente, partiremo da un'analisi di un modello base e semplificato, per poi spostarci su applicazioni più realistiche con l'aggiunta di segnali che possano modellare in maniera più verosimile delle casistiche reali.

3.1 Modello base

Riprendiamo quanto trattato nella Sezione 2.2. Come avevamo detto, un PSG si compone di:

- Due giocatori, un attaccante $\mathcal{A}$ e un difensore $\mathcal{D}$;
- Un grafo non orientato $G = (V, E)$, con V insieme dei vertici ed E insieme degli archi;
- Un sottoinsieme dei nodi del grafo, detto insieme dei Target $T \subseteq V$, che sono obiettivi sensibili per $\mathcal{A}$ e $\mathcal{D}$;
- Un valore $\pi(t) \in (0, 1]$, con $t \in T$ target, che rappresenta il valore del singolo target;
- Un tempo di penetrazione $d(t) \in \mathbb{N}$, con $t \in T$ target, che indica il numero di turni necessari per completare con successo un attacco al target t.

I due giocatori agiranno compiendo le loro mosse simultaneamente. Spiegando più dettagliatamente le azioni che i due giocatori possono effettuare, avremo che:

- L'attaccante $\mathcal{A}$, se non ha attaccato nei turni precedenti, può osservare le mosse del difensore $\mathcal{D}$, osservando la sua posizione e i suoi eventuali percorsi di pattuglia. Può quindi decidere se attaccare un qualsiasi target da lui deciso, oppure aspettare un turno. Considerando il caso pessimo, ipotizziamo che l'attacco ha inizio esattamente nell'istante in cui l'attaccante decide di sferrare il suo attacco.

- Il difensore $\mathcal{D}$, tuttavia, non ha nessuna informazione a proposito dell'attaccante $\mathcal{A}$. Non conosce infatti se attaccherà, nè potrà dedurre quando e dove questo attacco verrà sferrato. L'unica azione che può compiere è quindi scegliere quale vertice visitare il turno successivo. Può decidere di restare nello stesso vertice, oppure spostarsi in un nodo adiacente. Le mosse che egli effettua per muoversi sul grafo hanno un costo temporale che viene pagato con il movimento effettuato.

Il gioco termina in due possibili casi. Se l'attaccante viene scoperto dal difensore, ovvero se la guardia passa per il target t scelto dal malvivente dopo meno di $d(t)$ turni, bloccando quindi l'attacco e catturando $\mathcal{A}$. L'altro caso è quello in cui il target t scelto non viene pattugliato per più di $d(t)$ turni, consentendo all'attaccante di completare con successo l'attacco.

3.2 Introduzione di un sistema di allarmi

Per rendere questo gioco realistico, introduciamo l'utilizzo di un sistema di allarmi che genera segnali, con determinate caratteristiche.

Definizione 8. *Il sistema di allarmi è soggetto a falsi positivi e falsi negativi. E' inoltre spazialmente imperfetto.*

Per falsi positivi si intende quando, per qualsiasi motivo, un allarme viene attivato e il segnale è inviato al difensore nonostante non sia in corso alcun tipo di attacco. Per falsi negativi invece si intende la situazione in cui un attacco è in corso, ma a causa di un malfunzionamento del sistema di sensori l'allarme non notifica l'attacco in corso e quindi non viene inviato nessun segnale alla guardia. Per questa ricerca ci siamo focalizzati sull'analisi dei falsi negativi. Per spazialmente imperfetti si intende che se un attacco è in atto verrà notificato al difensore, tuttavia egli non conoscerà l'esatta posizione in cui l'attacco è in corso.

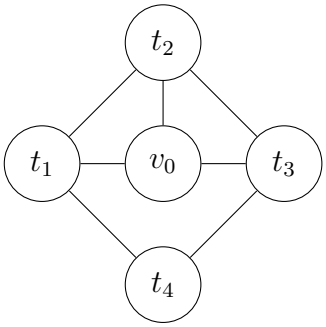
Ricapitolando, l'aggiunta dei segnali porta ad avere nel modello l'aggiunta di:

- Un insieme $S = s_1, s_2, ..., s_n$ di tutti i segnali presenti. Definiamo inoltre con $T(s_i)$ il supporto, ovvero i target coperti dal segnale s_i, mentre definiamo con $S(t_i)$ l'insieme dei segnali che coprono il target t_i;
- Una distribuzione di probabilità $p(s|t) \in [0, 1]$, che indica la probabilità che il segnale s venga attivato quanto il target t è sotto attacco;
- Un parametro $\alpha \in [0, 1]$, che rappresenta la probabilità di falso negativo. Più basso è questo valore, più il sistema di allarmi è accurato.

Chiaramente l'utilizzo di segnali porterà il difensore $\mathcal{D}$ a reagire prontamente a esso, recandosi il più rapidamente possibile nei luoghi in cui possa essersi verificata l'emergenza.

Per chiarire come è composto un grafo di gioco, e mostrare la distribuzione dei segnali, utilizziamo questo esempio.

Esempio 8. *In Figura 3.1 è presente un possibile grafo, con 5 nodi, di cui 4 target (t_1, t_2, t_3, t_4). In questo caso ci sono due segnali, s_1 e s_2, che*



t	$\pi(t)$	$d(t)$	$p(s_1\|t)$	$p(s_2\|t)$
t_1	0.58	4	0.8	0.2
t_2	0.9	4	0.65	0.35
t_3	0.63	4	0.4	0.6
t_4	0.35	4	0	1

Figura 3.1: Esempio di grafo.

coprono tutti i target, con le probabilità presenti in tabella. Notare che per ogni target, la somma delle probabilità dei segnali che lo coprono è sempre pari a 1.

3.2.1 Albero del gioco

Delineeremo qui l'albero del gioco del nostro modello, come rappresentato in Figura 3.2[1]. Questo albero può essere letto attraverso determinati step:

[1]Oltre alla notazione già spiegata, verranno aggiunti dei nodi $\mathcal{N}$ che indicano delle situazioni che avvengono con una certa probabilità p indipendente dai giocatori.

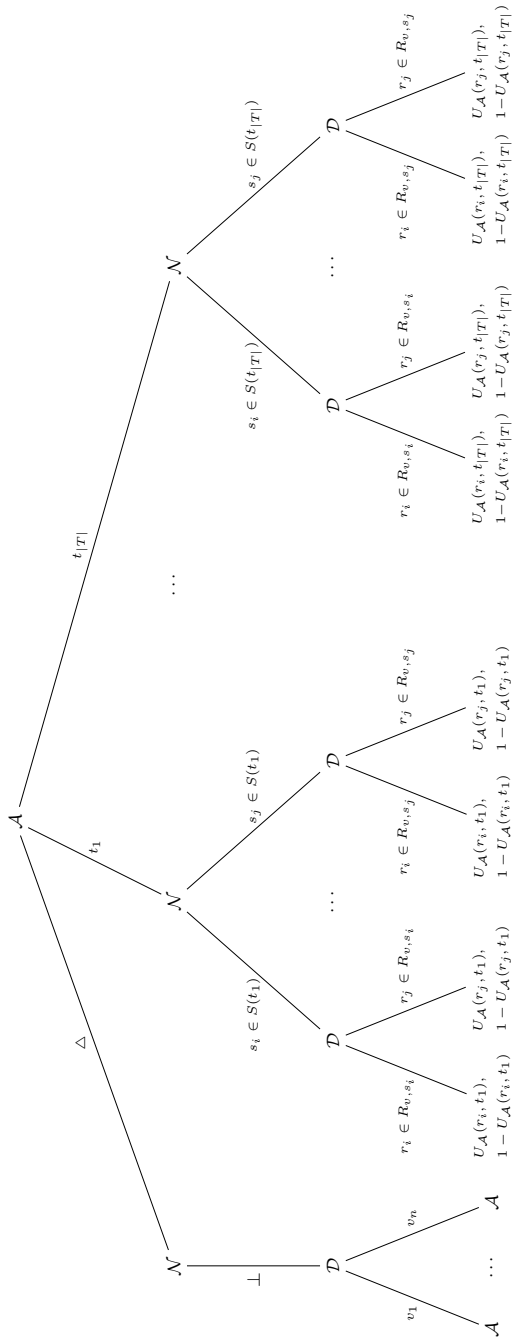


Figura 3.2: Albero di gioco, in cui assumiamo v come vertice corrente per $\mathcal{D}$. r è una sequenza di vertici chiamata *rotta*.

- *Radice dell'albero.* L'attaccante $\mathcal{A}$ decide se aspettare un turno (azione rappresentata con il simbolo $\triangle$), oppure se attaccare un target $t_i \in T$;
- *Secondo livello dell'albero.* In questo caso $\mathcal{N}$ indica il sistema di allarme, il cui comportamento è stato specificato a priori dalla distribuzione di probabilità $p(s|t)$ che determina il segnale generato a seconda dell'attacco di $\mathcal{A}$. I casi principali che possono verificarsi sono:
  1. Se nessun attacco è presente, nessun segnale verrà generato, ovvero $p(\perp|\triangle) = 1$, in quanto abbiamo ipotizzato nulla la probabilità di falso positivo;
  2. Se un attacco su un target t_i è in corso, il segnale *non* si attiverà con una probabilità $p(\perp|t_i) = \alpha$, ovvero la probabilità di falso negativo;
  3. Se un attacco su un target t_i è in corso, il segnale $s \in S(t_i)$ si attiverà con una probabilità $\overline{p}(s|t_i) = (1 - \alpha)p(s|t_i)$.
- *Terzo livello dell'albero.* Il difensore $\mathcal{D}$ osserva i segnali che vengono eventualmente attivati e decide il vertice seguente in cui spostarsi, tra quelli adiacenti al nodo attuale.
- *Quarto livello dell'albero.* In questo caso si possono verificare due casistiche:
  1. Se nessun attacco è presente, allora l'albero del sottogioco che parte da qui è identico all'albero dell'intero gioco, con l'unica differenza data dalla posizione iniziale di $\mathcal{D}$ che potrebbe essere differente da quella iniziale;
  2. Se un attacco è in corso, sia se è segnalato dal sistema di allarmi, sia se non è segnalato, solamente il difensore può eseguire delle azioni.

Come abbiamo potuto vedere, questo albero può essere suddiviso in sottoalberi ricorrenti, risolvibili separatamente.

3.2.2 Possibili stati del gioco

In relazione alle possibili azioni dell'attaccante, come delineato precedentemente, si può arrivare a due casistiche principali:

- *Se l'attaccante attacca il target t_i, il quale attiva il segnale $s_i \in S(t_i)$.* In questo caso $\mathcal{A}$ resterà impegnato per $d(t_i)$ turni nel tentativo di

completare l'attacco su quel target. Questo genererà un gioco per il difensore, il quale si troverà in un vertice v e dovrà rispondere al segnale per cercare di proteggere nel più breve tempo possibili tutti i target $T(s_i)$ coperti dal segnale (che ricordiamo spazialmente imperfetto, quindi a un segnale corrispondono più target). Chiamiamo questo gioco *Signal Response Game* da v, più abbreviato per comodità in SRG-v;

- *Se l'attaccante non effettua nessuna mossa, oppure se il segnale non si attiva per falso negativo.* Qui avremo che il difensore non ha nessun segnale a cui reagire, non sapendo se l'attaccante ha fatto la propria mossa o meno. Di conseguenza $\mathcal{D}$ deciderà la sua mossa ottimale tramite un gioco che chiamiamo *Patrolling Game*, abbreviato in PG. L'esito di questo gioco dipende direttamente dalle utilità dei vari nodi da pattugliare, che otterremo dalla risoluzione del SRG-v.

3.2.3 Giochi per il difensore

Come appena visto, il difensore si trova a dover affrontare due giochi, in base alle azioni dell'attaccante. Vediamo più nel dettaglio questi possibili casi.

Patrolling game

Quando nessun segnale è attivo, le scelte per il difensore sono sostanzialmente due: posizionarsi nel punto migliore e attendere, che chiameremo *best placement*, oppure pattugliare tramite un ciclo di pattugliamento, che chiameremo *covering cycle*.

- *Best placement.* Risolvendo il gioco di risposta ai segnali per ogni possibile vertice, noi conosciamo l'utilità per ognuno di essi. Possiamo perciò calcolare quale vertice è più vantaggioso come posizionamento iniziale per il difensore, che può quindi attendere nella posizione fino allo scattare di un segnale. Tuttavia questa scelta si rivela poco efficace nel caso di falsi negativi, siccome in quel caso l'attaccante potrà agire totalmente indisturbato. Bisogna ricordare che un falso negativo coincide con l'assenza dell'invio del segnale alla guardia, che in questo caso coincide con un attacco completato con successo da parte del ladro.

- *Covering cycle.* In questo caso il problema diventa più complesso, in quanto il difensore continuerà a spostarsi in un ciclo di pattugliamento,

rispondendo ai possibili segnali ricevuti. Ipotizzando la possibilità di falsi negativi può quindi proteggere un maggior numero di target grazie al continuo spostamento tra essi durante il proprio percorso. Questo è dato dal fatto che l'intervallo di tempo tra due visite successive dei target è sempre minore al loro tempo di penetrazione, garantendo quindi la loro protezione.

Signal response game

In caso di SRG-v, l'obiettivo del difensore è quello di trovare la miglior strategia per rispondere a ogni segnale s partendo dal vertice v. Bisogna notare che tutti i SRG-v sono indipendenti tra loro, pertanto la miglior strategia di uno di questi giochi non dipende dalle strategie degli altri. Risolvendo questo tipo di giochi possiamo far collassare tutta la sequenza di mosse compiute da $\mathcal{D}$ in una singola azione, vista come risposta alla decisione di $\mathcal{A}$ di attaccare un determinato target (e quindi di attivare un preciso segnale).

Capitolo 4

Algoritmi

"La teoria è quando si sa tutto e niente funziona.
La pratica è quando tutto funziona e nessuno sa il perché.
Noi abbiamo messo insieme la teoria e la pratica: non c'è niente che funzioni... e nessuno sa il perché!"

Albert Einstein

In questa sezione discuteremo dei vari algoritmi utilizzati e delle euristiche per risolvere i vari giochi.

4.1 Struttura della soluzione

Per risolvere il gioco come prima cosa semplifichiamo il grafo $G = (V, E)$ in un grafo $G' = (V', E')$ in cui i vertici V' saranno tutti e soli i target, e con E' indicheremo, se presente, il cammino più breve tra due obiettivi senza passare per altri target.

Esempio 9. *Riprendiamo qui il grafo in Figura 3.1. Come possiamo notare è presente un vertice non target, v_0, che provvederemo a rimuovere. In aggiunta, avremo dei nuovi archi, in quanto esiste un percorso tra il vertice t_1 e il vertice t_3 che non passa per nessun target. Aggiungeremo pertanto tale collegamento nel grafo, come mostrato in Figura 4.1. Il collegamento presente tra il vertice v_0 e il vertice t_2 ad esempio verrà semplicemente rimosso, in quanto non porta t_2 ad avere un collegamento verso nuovi target, nè il suo utilizzo riesce a ridurre il costo di qualche arco già presente.*

Bisogna tuttavia prestare attenzione ai costi degli archi aggiunti, i quali avranno come peso la somma dei costi del cammino più breve tra i due target che esso collega.

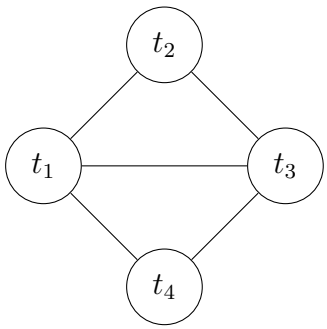


Figura 4.1: Esempio di grafo da pattugliare.

4.1.1 Rotte di pattugliamento

Come detto nella parte di definizione del problema3.2.3, essendo i sensori soggetti a falsi negativi l'approccio del best placement non è quello più efficace. Avremo pertanto bisogno di trovare un percorso di pattugliamento che massimizzi l'utile del difensore. Chiariamo pertanto il concetto di Covering Cycle.

Definizione 9. *Si definisce Covering Cycle un insieme di target tale per cui:*

- *Nodo iniziale e nodo finale del ciclo coincidono;*
- *I target appartenenti al ciclo non vengono mai lasciati senza essere sorvegliati per più di $d(t)$ turni.*

Dato un covering cycle, si può definire anche il concetto di covering set.

Definizione 10. *Un Covering Set è un sottoinsieme di target $T(c)$ tale da avere c come covering cycle.*

La differenza fondamentale tra covering cycle e covering set è che un covering cycle è una sequenza ordinata di nodi che indica l'ordine con cui devono essere effettivamente visitati, mentre un covering set è un insieme non ordinato di nodi.

Possiamo ipotizzare per semplicità che a un determinato insieme di target corrisponda uno e un solo ciclo di copertura passante per questi vertici.

Pertanto il difensore $\mathcal{D}$, adottando il covering cycle c proteggerà i target $T(c)$ catturando l'attaccante se esso attaccherà uno di questi obiettivi, rendendoli sicuri anche nel caso di falso negativo.

4.1.2 Istanze di gioco

Ottenuto quindi il grafo ridotto, analizziamo l'istanza $\mathcal{I}$ del gioco che il nostro algoritmo riceverà in ingresso.
I principali dati che il nostro algoritmo necessiterà sono:

- *Matrice di adiacenza.* Per conoscere se tra due vertici del grafo esiste un arco che li collega;
- *Matrice pesata degli archi.* Rappresenta il costo degli archi del grafo;
- *Valori dei targets.* Contiene i valori $\pi(t)$ per ogni target t;
- *Deadlines dei targets.* L'insieme delle deadlines $d(t)$ di ogni target t;
- *Segnali.* Per ogni possibile segnale, rappresenta la probabilità che si attivi nel caso in cui il target t è sotto attacco;
- *Probabilità di falso negativo.* Probabilità $\alpha \in [0, 1]$ che si verifichi un falso negativo.

Queste istanze verranno create con dati realistici, in modo da rappresentare l'area mappata nel modo migliore.

4.1.3 Panoramica della soluzione

Ricevuta l'istanza di gioco, l'approccio risolutivo può essere sintetizzato nel seguente modo, come spiegato in Figura 4.2:

1. Selezione di un subset di targets T' tramite metodi euristici;
2. Interrogazione di un Covering Cycle Oracle (CCO), che restituisce un covering cycle C su T' se esiste;
3. Se il ciclo esiste, interrogazione di un Signal Response Oracle (SRO) che calcola le strategie ottimali di risposta ai segnali;
4. Salvare la strategia e il suo valore in una lista;
5. Ripetere dallo step 1.

Analizziamo ora tutte le componenti di questo algoritmo nel dettaglio.

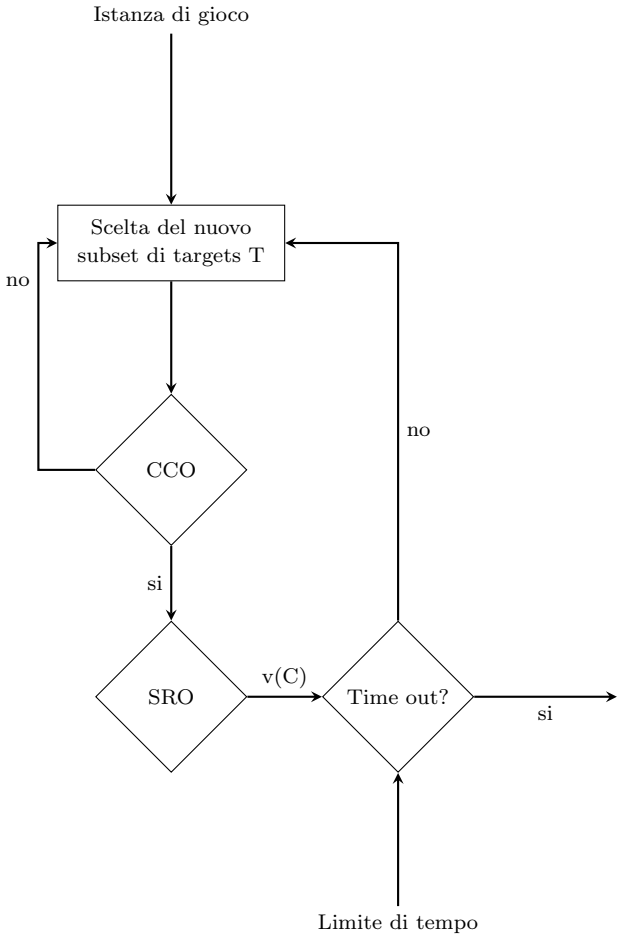


Figura 4.2: Approccio risolutivo.

4.2 Euristiche

Discutiamo in questa sezione tutte le euristiche impiegate nella selezione dei subset di targets T' da analizzare. Proponiamo qui tre principali euristiche, le quali si occuperanno di ottimizzare la generazione di tutti i possibili subset, aumentando quindi la probabilità di eseguire fin da subito l'algoritmo sui gruppi di targets più interessanti.

Le euristiche qui esposte si dividono in due statiche (una basata sul valore dei targets e una sulla loro distanza) e una dinamica (basata sulle risposte dell'attaccante).

Un'euristica si dice *statica* se dipende solo dall'istanza del problema e non si modifica l'ordine dei subset di target considerati. In caso contrario prende il nome di euristica *dinamica*.

4.2.1 Metodo statico basato sul valore dei targets

Questo metodo utilizza il valore $\pi(t)$ per stimare quanto può essere vantaggioso inserire un target nel covering set da analizzare. Più precisamente, daremo priorità ai sottoinsiemi con un alto valore cumulativo e allo stesso tempo con pochi elementi. Andremo a espandere questi gruppi di targets, di cui analizzeremo ogni possibile enumerazione, aggiungendo di volta in volta gli obiettivi più interessanti ordinati in base al loro valore.

Questa euristica si basa sull'ipotesi di egoismo tra giocatori, che renderà sia attaccante sia difensore maggiormente attratti da obiettivi con un alto $\pi(t)$.
In pratica, questa euristica è caratterizzata dai seguenti passaggi:

1. Ordinare i targets in relazione al proprio valore $\pi(t)$, suddividerli in sottoinsiemi $T^{(i)}$ che contengono i gruppi di targets con l'i-esimo valore più alto e infine assegnare $U = T^{(1)}$ e $u = 1$;

2. Enumerare tutti i possibili sottoinsiemi Q, dove $Q \subseteq U$ e, se $u > 1$, $Q \nsubseteq \bigcup_{i \in \{1,\dots,u-1\}} U^i$; per ognuno di questi, si computa σ eseguendo i due oracoli (CCO e SRO);

3. Impostare $U = U \bigcup U^{(u+1)}$, $u = u + 1$ e ripetere dallo step 2.

Cerchiamo di rendere più chiara questa spiegazione con un esempio.

Esempio 10. *Riproponiamo il grafo in Figura 4.1. Qui sono presenti 4 nodi, ipotizziamo con valore* 0.8 *per i nodi* 1 *e* 2*,* 0.6 *per il nodo* 3 *e* 0.9 *per il nodo* 4*. Otteniamo come prima cosa un ordinamento dei targets dato dal loro $\pi(t)$, che corrisponde a $T^{(1)} = \{4\}, T^{(2)} = \{1, 2\}$ e infine $T^{(3)} = \{3\}$. A*

questo punto, alla prima iterazione, avremo solamente effettuato l'analisi sul covering set $\{4\}$*. Nella seconda iterazione però, avremo che* $U = \{1, 2, 4\}$ *e quindi enumereremo tutti i possibili sottoinsiemi di* U *non ancora analizzati. Otteniamo quindi* $Q = \{\{1\}, \{2\}, \{1, 2\}, \{1, 4\}, \{2, 4\}, \{1, 2, 4\}\}$*, sui quali eseguiremo i due oracoli. Per concludere, l'ultima iterazione verrà effettuata aggiungendo il nodo* 3*, ovvero sui sottoinsiemi*

$$Q = \{\{3\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{1, 2, 3\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 4\}\}$$

4.2.2 Metodo statico basato sulla distanza tra i targets

Vediamo ora un'altra euristica statica basata in questo caso sulla distanza, data dal peso degli archi, tra due vertici del grafo. Questo metodo cercherà infatti di privilegiare i sottoinsiemi di targets dove la distanza più breve tra ogni coppia di nodi inclusi è sempre inferiore a una data soglia.

Questa euristica si basa sull'idea che i gruppi di nodi più vicini tra loro siano più facilmente copribili da un covering cycle, in quanto richiedono meno tempo per essere raggiunti.
Vediamo i passaggi che caratterizzano questo metodo:

1. Fissare un valore di soglia $\tau = 1$;
2. Enumerare tutti i sottoinsiemi di targets T' tali per cui per ogni $t_1, t_2 \in T'$ si verifica che $d(t_1, t_2) \leq \tau$;
3. Ripetere dallo step 2 dopo aver impostato $\tau = \tau + 1$.

Chiariamo anche qui il concetto con un esempio.

Esempio 11. *Prendiamo nuovamente il grafo presente in Figura 4.1. Il costo degli archi di questo grafo (e quindi la distanza tra i targets) è pari a* 2 *tra i nodi* 1 *e* 3*, mentre è pari a* 1 *in tutti gli altri collegamenti. Dopo aver inizializzato* $\tau = 1$*, enumeriamo tutti i sottoinsiemi di targets la cui distanza è inferiore a* τ*. Otterremo quindi*

$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{3, 4\}, \{1, 2, 3\}, \{1, 3, 4\}\}$$

Alla seconda iterazione, dopo aver incrementato $\tau = \tau + 1 = 2$*, avremo come ultimo sottoinsieme di targets da analizzare*

$$\{\{2, 4\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\}$$

Nella pratica, andremo di volta in volta a rimuovere tutti gli archi con un peso superiore al valore di soglia, mentre aggiungeremo nuovi collegamenti

tra tutti i vertici che hanno un percorso che li collega con costo inferiore a τ. Del grafo ottenuto andremo a ottenere le *cricche* massimali e tutte le loro sotto-cricche, che saranno quindi i sottoinsiemi di targets da analizzare.

Definizione 11. *Una cricca (o clique) è un insieme di vertici V tali per cui, per ogni coppia di vertici $(v_1, v_2) \in V$, esiste un arco che li collega. Una cricca si dice massimale se non esiste una cricca che la contiene.*

4.2.3 Metodo dinamico basato sulle risposte dell'attaccante

In questo caso, l'ordinamento viene calcolato in base alle strategie dell'attaccante, aggiornandola di volta in volta. Con questo procedimento cercheremo di includere progressivamente i targets che saranno strategicamente più interessanti per $\mathcal{A}$. Sinteticamente vorremmo ottenere una situazione tale per cui, partendo dall'insieme T' a cui corrisponde una strategia σ', possiamo costruire T'' includendo in T' i target che offrono una maggiore utilità all'attaccante dato σ' e ottenere da esso una miglior strategia σ''.

Cercheremo quindi di partire dal miglior posizionamento statico, aggiungendo poi nuovi obiettivi considerando le azioni di $\mathcal{A}$ ordinate in base alla loro utilità attesa.

In questo caso, l'idea su cui ci basiamo è che l'attaccante osserva le mosse del difensore e modifica le proprie strategie. Il difensore andrà di conseguenza ad adattarsi alle mosse di $\mathcal{A}$, cercando di coprire i target da lui più ambiti.
L'algoritmo è riassumibile nei seguenti passaggi:

1. Computare la strategia ottimale σ trovando un posizionamento statico w^* e impostare $U = \{w^*\}$;

2. Ordinare i targets non inclusi in U in base alla miglior utilità attesa da $\mathcal{A}$ attaccandoli. Costruire poi $T^{(i)}$ come insieme di obiettivi dove un attacco garantirebbe all'attaccante l'i-esimo valore maggiore;

3. Per $i = 1, 2, ...$ enumerare i sottoinsiemi Q_i, dove $Q_i \subseteq U \bigcup U^{(i)}$ e $Q_i \nsubseteq U$; per ognuno di questi, computare σ eseguendo i due oracoli;

4. Ripetere dallo step 2 per ogni strategia σ computata impostando $U = Q_i$.

Proponiamo anche qui un esempio per chiarire il concetto.

Esempio 12. *Applichiamo l'algoritmo al grafo in Figura 4.1. Come già detto, qui sono presenti 4 nodi, con valore* 0.8 *per i nodi* 1 *e* 2, 0.6 *per il nodo*

3 *e* 0.9 *per il nodo* 4. *Il miglior posizionamento per il difensore è nel nodo* 4, *in quanto è il più appetibile per entrambi i giocatori. A questo punto, gli altri target verranno ordinati in base all'utilità attesa dell'attaccante, in questo caso* $2, 1$ *e infine* 3, *per poi creare delle coppie con il target* 4 *e eseguendo gli oracoli su ognuno di questi sottoinsiemi. Saranno quindi analizzate le coppie*

$$\{\{2,4\},\{1,4\},\{3,4\}\}$$

Da queste coppie verrà ripetuto l'algoritmo più volte. Per semplicità espandiamo solo il caso $\{2,4\}$, *gli altri casi procederanno in modo analogo. In questo caso i due target rimanenti sono ordinati avendo prima* 1 *e poi* 3. *I gruppi generati e poi analizzati sono quindi*

$$\{\{1,2,4\},\{2,3,4\}\}$$

Per poi arrivare a concludere con l'inserimento dell'ultimo target nel sottoinsieme $\{1,2,3,4\}$

Per applicarlo, utilizzeremo un metodo per il ranking dei subset di targets, cosi da analizzare prima quelli più interessanti. Per far ciò ci avvaleremo dell'uso di una particolare euristica che, dopo la designazione del *best placement*, assegnerà un valore di "interesse" al sottoinsieme in analisi.
Più nel dettaglio, assegneremo alle coppie di target l'euristica

$$h = \begin{cases} 0, & \text{se i target } t_1 \text{ e } t_2 \text{ sono entrambi di interesse per l'attaccante} \\ 1, & \text{se uno e un solo target tra } t_1 \text{ e } t_2 \text{ è di interesse per l'attaccante} \\ 2, & \text{se i target } t_1 \text{ e } t_2 \text{ non sono di interesse per l'attaccante} \end{cases}$$

Quando a un gruppo di target di cui conosciamo già il valore dell'euristica aggiungiamo un nuovo vertice, il nuovo risultato sarà ottenuto sommando al precedente il valore

$$h = \begin{cases} 0, & \text{se il nuovo target } t \text{ è di interesse per l'attaccante} \\ 1, & \text{se il nuovo target } t \text{ non è di interesse per l'attaccante} \end{cases}$$

Con questo metodo potremo infatti effettuare un ordinamento sui sottoinsiemi di obiettivi, definendo quale gruppo di targets analizzare per primo.

4.3 Covering Cycle Oracle

Il CCO, come mostrato precedentemente, riceve in ingresso un sottoinsieme di targets T' e si occupa di cercare se ne esiste un ciclo di copertura. L'oracolo darà in uscita questo covering cycle se trovato, nulla altrimenti.

Analizziamo ora più nel dettaglio ingressi e uscite di questa struttura, per poi passare ad esaminarne l'algoritmo.

Ingressi e uscite

Per eseguire questo oracolo, avremo bisogno di acquisire in ingresso:

- L'insieme dei targets T, che rappresenteranno i nodi del grafo da analizzare;
- La matrice di adiacenza A, che consentirà di conoscere se due nodi del grafo sono collegati direttamente da un cammino;
- La matrice pesata di adiacenza w, per conoscere i pesi degli archi che collegano due vertici;
- Le deadline per ogni target d, per sapere quanto tempo un obiettivo necessita per essere sottratto.

In uscita invece, se esiste, avremo una sequenza σ di targets tale per cui:

- La sequenza σ è ciclica, ovvero il primo e l'ultimo vertice coincidono;
- Ogni target in T è visitato almeno una volta, ovvero non ci sono vertici non coperti da σ;
- Se il ciclo viene ripetuto indefinitamente, per ogni $i \in T$, l'intervallo di tempo tra due visite successive di i non è mai superiore a $d(i)$.

Struttura algoritmica

Formalizziamo ora come, partendo dall'ingresso sopra descritto, arriveremo a ottenere l'uscita desiderata.
Per far ciò introduciamo prima un po' di notazione necessaria.

- La sequenza σ utilizzata in precedenza la definiamo come una funzione $\sigma : \{1, 2, ..., s\} \to T$, dove $\sigma(j)$ è il j-esimo elemento della sequenza e s la sua lunghezza, non nota a priori;
- Indichiamo con $O_i(j)$ la posizione in σ della j-esima occorrenza del vertice i;
- Con o_i mostriamo il numero totale di occorrenze dell'elemento i in una data σ.

Ad esempio, data $\sigma = \{1, 2, 3, 2, 1\}$, $O_2(1) = 2$ e $O_2(2) = 4$, mentre $o_2 = 2$ e $o_5 = 0$.

Avremo quindi che un ciclo di copertura σ per il difensore sarà dato dalla soluzione di:

$$\sigma(1) = \sigma(s) \tag{4.1}$$

$$o_i \geq 1 \quad \forall i \in T \tag{4.2}$$

$$a'\left(\sigma(j-1), \sigma(j)\right) = 1 \quad \forall j \in \{2, 3, \ldots, s\} \tag{4.3}$$

$$\sum_{j=O_i(k)}^{O_i(k+1)-1} w\left(\sigma(j), \sigma(j+1)\right) \leq d(i) \quad \forall i \in T, \forall k \in \{1, 2, \ldots, o_i - 1\} \tag{4.4}$$

$$\sum_{j=1}^{O_i(1)-1} w\left(\sigma(j), \sigma(j+1)\right) + \sum_{j=O_i(o_i)}^{s-1} w\left(\sigma(j), \sigma(j+1)\right) \leq d(i) \quad \forall i \in T \tag{4.5}$$

Per chiarire il significato di questo problema, abbiamo:

- In 4.1 imponiamo che sia un ciclo, ovvero il primo e l'ultimo vertice devono coincidere;
- In 4.2 imponiamo che ogni nodo sia visitato almeno una volta;
- In 4.3 diciamo che due vertici consecutivi nel ciclo di pattugliamento devono essere connessi direttamente da un arco;
- In 4.4 imponiamo che, se in un singolo ciclo un nodo è visitato più di una volta, l'intervallo di tempo è inferiore a $d(t)$;
- In 4.5 imponiamo che, per ogni vertice, l'ultima visita di un ciclo e la prima del ciclo successivo sia in un intervallo di tempo inferiore a $d(t)$.

L'algoritmo risolutivo, presentato in [1], è presente qui di seguito.

L'Algoritmo 1 assegna a $\sigma(1)$ un vertice casuale. Siccome la soluzione è un ciclo che visita ogni vertice, questo assegnamento non pregiudica la possibilità di trovare una soluzione.

Algoritmo 1 Ricerca di un covering cycle

```
1: procedure FindSolution(T, A, w, d)
2:     σ(1) ← Random(T)
3:     σ ← RecursiveCall(T, A, w, d, σ, 2)
4:     return σ
5: end procedure
```

L'algoritmo 2 assegna a $\sigma(j)$ un vertice appartenente a $F_j \subseteq T$, che contiene i valori possibili di $\sigma(j)$ calcolati dall'algoritmo di forward checking. L'algoritmo termina se F_j è vuoto o nessun vertice in F_j può essere assegnato a $\sigma(j)$.

Algoritmo 2 Ricerca di un covering cycle: chiamata ricorsiva

```
1: procedure RecursiveCall(T, A, w, d, σ, iteration)
2:     if σ(1) = σ(iteration − 1) and costraint (4.2) verificato then
3:         if costraint (4.5) verificato then
4:             return σ
5:         else
6:             return ∅
7:         end if
8:     else
9:         F_iteration ← ForwardChecking(T, A, w, d, σ, iteration)
10:        for all t ∈ F_iteration do
11:            σ(iteration) ← t
12:            σ' ← RecursiveCall(T, A, w, d, σ, iteration + 1)
13:            if σ' ≠ ∅ then
14:                return σ'
15:            end if
16:        end for
17:        return ∅
18:    end if
19: end procedure
```

L'algoritmo 3 restringe F_j a tutti i nodi che sono raggiungibili direttamente dall'ultimo vertice assegnato $\sigma(j-1)$ e che la loro visita non violi i vincoli 4.4 e 4.5. In questo algoritmo denotiamo con $\overline{w}$, ad esempio $\overline{w}(i,j)$, il peso del cammino più breve tra i due vertici i e j.
Vediamo ora un esempio di come questo algoritmo verrebbe applicato.

Esempio 13. *Anche in questo caso, utilizziamo il grafo di Figura 4.1. Come valori, deadline e costi assegnamo, per semplicità, un valore di* 1 *in tutti i targets (utilità ripartita equamente), deadlines pari a* 4 *per ognuno e come costi degli archi assegnamo* 2 *nel collegamento tra i nodi* 1 *e* 3 *mentre* 1 *a tutti gli altri. Avviando l'algoritmo, assumiamo per semplicità che il nodo iniziale scelto randomicamente sia* 1 *(come già discusso non modifica la soluzione). Viene eseguito per la prima volta il forward checking, che restituisce* $F_j = \{2,3,4\}$. *Scegliamo ad esempio il primo vertice, ottenendo quindi* $\sigma = \{1,2\}$. *Da* 2*, eseguiamo nuovamente il forward checking, per*

Algoritmo 3 Ricerca di un covering cycle: forward checking

procedure ForwardChecking(T, A, w, d, σ, iteration)
  $F_{iteration} \leftarrow \emptyset$
  $s \leftarrow iteration - 1$
  for all $i \in T$ adiacenti a $\sigma(s)$ **do**
    if valgono le seguenti condizioni:
$\Big(o_i = 0 \wedge \sum_{l=1}^{s-1} w(\sigma(l), \sigma(l+1)) + w(\sigma(s), i) + \overline{w}(i, \sigma(1)) \leq d(i)$ or
$o_i > 0 \wedge \sum_{l=O_i(o_i)}^{s-1} w(\sigma(l), \sigma(l+1)) + w(\sigma(s), i) \leq d(i)\Big)$ and,
for all $k \neq i$,
$\Big(o_k = 0 \wedge \sum_{l=1}^{s-1} w(\sigma(l), \sigma(l+1)) + w(\sigma(s), i) + \overline{w}(i, k) + \overline{w}(k, \sigma(1)) \leq d(k)$ or
$o_k > 0 \wedge \sum_{l=O_k(o_k)}^{s-1} w(\sigma(l), \sigma(l+1)) + w(\sigma(s), i) + \overline{w}(i, k) \leq d(k)\Big)$
      then
        $F_{iteration} \leftarrow F_{iteration} \cup i$
      end if
  end for
  return $F_{iteration}$
end procedure

ottenere $F_j = \{1, 3\}$. *Se scegliessimo il target* 1 *ci troveremmo allo stesso punto dell'inizio, con la differenza che non potremmo coprire tutti i nodi in tempo. Scegliamo quindi il vertice* 3. *Da* $\sigma = \{1, 2, 3\}$, *il forward checking restituisce* $F_j = \{1, 2, 4\}$. *Come prima, se scegliamo* 1 *o* 2 *avremmo in seguito che il forward checking restituirebbe un insieme vuoto, in quanto le deadlines non sarebbero rispettate. Prendiamo quindi come nuovo elemento* 4. *L'ultimo forward checking restituirà il solo elemento* 1, *che aggiuneremo, ottenendo* $\sigma = \{1, 2, 3, 4, 1\}$ *come ciclo di copertura.*

Complessità algoritmica

Parliamo ora della complessità dell'algoritmo sopra esposto. Come vediamo, a ogni iterazione possiamo aggiungere, nel caso pessimo $|T| - 1$ nodi. La complessità diventa dunque $O(|T|^{k|T|})$, quindi una complessità polinomiale. La dimostrazione di questo gioco, in cui otteniamo il covering cycle dato un set di targets, è presente in [11].

4.4 Signal Response Oracle

Il SRO ha il compito di, preso in ingresso un ciclo di copertura C, computare la strategia ottimale di risposta ai segnali per ogni vertice del ciclo. Questo

problema può essere risolto separatamente per ogni target, basandoci sul fatto che la strategia del difensore è considerata fissata e pari al ciclo di pattugliamento.

Ingressi e uscite

Per l'esecuzione di questo oracolo, in input daremo necessariamente:

- L'insieme dei targets T, che rappresenteranno i nodi del grafo da analizzare;
- La matrice di adiacenza A, che consentirà di conoscere se due nodi del grafo sono collegati direttamente da un cammino;
- La matrice pesata di adiacenza w, per conoscere i pesi degli archi che collegano due vertici;
- Le deadline per ogni target d, per sapere quanto tempo un obiettivo necessita per essere sottratto;
- I valori di ogni target π, per conoscere quanto un obiettivo è importante da proteggere;
- La probabilità di falso negativo α, in modo tale che possiamo considerare nei calcoli il fatto che un allarme possa non attivarsi;
- Il ciclo di copertura C, come calcolato dal CCO, che coinciderà alla strategia di pattugliamento del difensore.

In questo caso l'uscita sarà l'utilità del difensore relativa a tale ciclo. Per ottenere questo valore, per ogni $t \in C$, è necessario costruire e risolvere la matrice del gioco di risposta ai possibili segnali. Di questi risultati utilizzeremo il valore più basso, in quanto caso pessimo, come utilità del ciclo di copertura.

Questo algoritmo verrà infatti eseguito su tutti i possibili cicli di pattugliamento del grafo, mostrandone l'utilità, per poi ricavarne il migliore.

Struttura algoritmica

Analizziamo ora come il signal response oracle è in grado di calcolare l'utilità di un dato ciclo C.

Il lavoro principale di questo oracolo è legato alla creazione della matrice di gioco, che dovrà essere generata e risolta per ogni vertice di C. Analizziamo ora come questa matrice è creata, procedendo per step da un caso base, fino a giungere a quella necessaria per il nostro problema.

Il caso più semplice possibile che trattiamo è quello in cui vi è un singolo segnale e assenza di falso negativo. Per rappresentare ciò, utilizzeremo una matrice indicizzata dai target del ciclo e dalle rotte di pattugliamento possibili per rispondere ai segnali.

	t_1	t_2	t_3
r_1			
r_2			
r_3			

Figura 4.3: Matrice con singolo segnale.

La struttura della matrice, presente in Figura 4.3, verrà poi popolata con:

$$U_{\mathcal{D}}(r,t) = \begin{cases} 1, & t \in r \\ 1-\pi(t), & t \notin r \end{cases}$$

Tuttavia questa semplice matrice, per come è costruita e popolata, non può gestire le casistiche più complesse. Come passo successivo aggiungiamo, sempre alla matrice in Figura 4.3, la possibilità di avere dei falsi negativi con probabilità α. Per far ciò possiamo lasciare identica la struttura matriciale, modificando solamente i valori inseriti, avendo

$$U_{\mathcal{D}}(r,t) = \begin{cases} (1-\alpha)+\alpha, & t \in r, t \in C \\ (1-\alpha)+\alpha(1-\pi(t)), & t \in r, t \notin C \\ (1-\alpha)(1-\pi(t))+\alpha, & t \notin r, t \in C \\ (1-\alpha)(1-\pi(t))+\alpha(1-\pi(t)), & t \notin r, t \notin C \end{cases}$$

Arriviamo ora al caso più complesso, in cui introduciamo la possibilità di avere più segnali. In Figura 4.4 vediamo come la matrice viene modificata,

		t_1	t_2	t_3	t_4
s_1	r_1				
	r_2				
	r_3				
s_2	r_1				
	r_2				
	r_3				
	s_0				

Figura 4.4: Matrice con segnali multipli.

inserendo delle rotte differenti per ogni segnale. Pertanto, la matrice viene

suddivisa in tante sottomatrici, una per ogni possibile segnale. Introduciamo inoltre il segnale s_0, che rappresenta l'assenza di segnale. In tal modo possiamo modellare qualsiasi caso possibile, inserendo poi i valori

$$U_{\mathcal{D}}(r_{s,i}, t) = \begin{cases} 1, & t \in r_{s,i} \\ 1 - \pi(t), & t \notin r_{s,i} \end{cases}$$

La sottomatrice del segnale nullo invece avrà i valori

$$U_{\mathcal{D}}(C, t) = \begin{cases} 1, & t \in C \\ 1 - \pi(t), & t \notin C \end{cases}$$

A questa matrice di gioco ne verrà aggiunta una nuova, detta matrice dei segnali.

	t_1	t_2	t_3
s_1			
s_2			
s_0			

Figura 4.5: Matrice dei segnali.

La struttura è mostrata in Figura 4.5. I valori contenuti in essa sono dati da $p(s|t)$, indicando la probabilità che il segnale s si attivi quando il target t è sotto attacco. I valori associati al segnale nullo saranno invece tutti 1, in quanto quel segnale sarà associato totalmente a ogni target.

In quest'ultima matrice avremo la gestione del falso negativo, in quanto moltiplicheremo tutti i valori contenuti in essa per $(1 - \alpha)$ se associati a un segnale s_i con $i > 0$. Nel caso del segnale nullo s_0 invece moltiplicheremo i valori con α.

Risolveremo il gioco tramite queste due matrici, ottenendo l'utilità del gioco dal vertice v relativa al ciclo C. Essa verrà salvata e confrontata con tutti i risultati trovati per ogni nodo. Il valore minimo sarà mantenuto come utilità del ciclo di pattugliamento C.

Complessità algoritmica

In questo caso, l'algoritmo calcola le rotte, popola la matrice del gioco e la risolve. La complessità del gioco dipende quindi dal numero di target nel ciclo di copertura C, dal numero totale di target T e dal numero di segnali S. Il calcolo più complesso in questa parte sarà dato dall'ottenimento delle rotte, svolto per ogni target del ciclo, per ogni target nel grafo, per ogni

segnale esistente. Avremo quindi che questa soluzione algoritmica presenterà una complessità $O(|C| * |T| * |S|)$.

Capitolo 5

Valutazioni sperimentali

"I nostri successi e i nostri fallimenti sono tra loro inscindibili,
proprio come la materia e l'energia.
Se vengono separati, l'uomo muore."

Nikola Tesla

E' arrivato il momento di vedere nella pratica il nostro modello, per capire se è realmente utilizzabile.

5.1 Istanze utilizzate

Sappiamo ormai come il problema è stato modellato e implementato, ma prima di descrivere gli esperimenti veri e propri è necessario discutere a proposito delle istanze che verranno utilizzate.

Chiaramente creare un insieme di istanze significative è un compito non troppo semplice, siccome le istanze possono risultare esageratamente semplici o complesse. In entrambi i casi produrrebbero dei risultati poco interessanti, in quanto se utilizzassimo casi semplicistici non testeremmo le effettive capacità del modello, mentre con istanze troppo complesse potremmo solo sprecare tempo senza ottenere risultati.

Utilizzando istanze completamente casuali ci porterebbe ad avere quasi sempre casi poco significativi, pertanto diventa necessario utilizzare dei vincoli che possano imporre un valore fisso ad alcuni parametri, mentre diano una scelta casuale ad altri.

Tutto questo riflette il fatto che la difficoltà principale è data dall'ottenere un set di istanze che rappresentino in maniera rilevante i casi realistici, ottenendo quindi delle mappe che possano essere il più possibile verosimili.

Le istanze saranno costruite così:

- Nel grafo $G = (V, E)$ ogni vertice sarà un target, in modo da semplificare l'analisi;
- Ogni arco del grafo avrà costo unitario;
- Tutte le deadlines sono fissate a $n - 1$, dove n è il numero di nodi del grafo;
- La probabilità di falso negativo α è sempre inferiore a 50%, per evitare di avere sensori fin troppo irrealistici;
- Ogni target sarà coperto da un solo segnale.

I restanti dati verranno scelti in maniera casuale.

Le istanze verranno presentate seguendo tre principali valori di riferimento, ovvero il numero di targets, la densità d'arco del grafo data da $\frac{\text{mod } E}{n(n-1)/2}$ e la probabilità di falso negativo α.

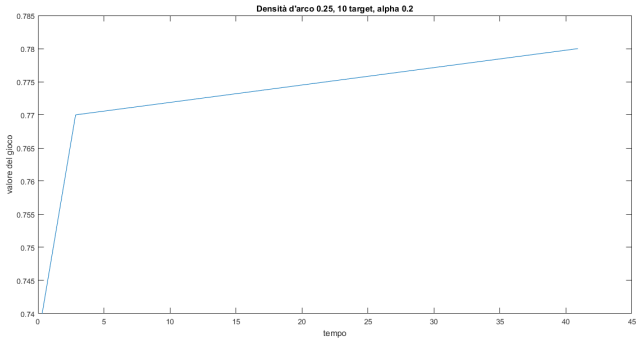
Per quanto riguarda l'implementazione, tutto il codice è stato implementato in MATLAB, sia per eseguire gli algoritmi, sia per creare istanze e grafici. Tutti gli esperimenti sono stati eseguiti su macchina Windows 4-core, con CPU da 2.4 GHz e 4 GB di RAM.

5.2 Risoluzione del gioco

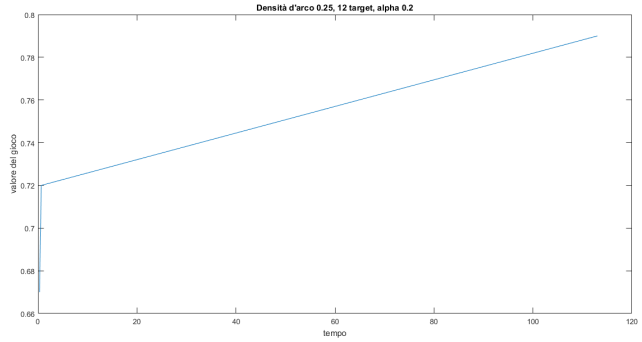
Procediamo ora a varie sperimentazioni per verificare come si comporta l'algoritmo sotto vari aspetti. Teniamo ben presente che il codice creato per risolvere gli algoritmi presentati in precedenza è *any-time*, ovvero è creato in modo tale che l'esecuzione può essere terminata dall'utente in qualunque momento del processo, ottenendo in uscita la miglior soluzione trovata fino a quell'istante. Questo metodo è pensato per istanze particolarmente difficili, che richiederebbero troppo tempo per essere computate e di cui accettiamo una soluzione sub-ottima che sia, in base a quanto tempo l'algoritmo verrà eseguito, più accurata possibile.

5.2.1 Tempo impiegato

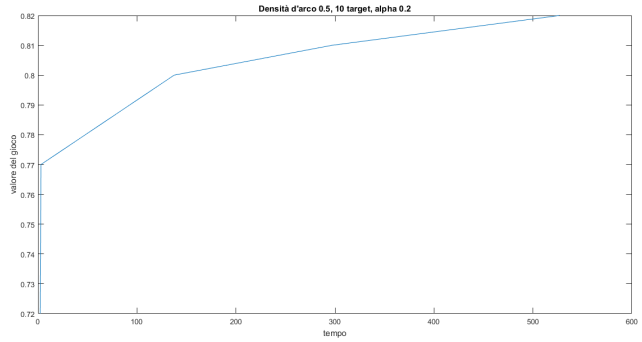
Come prima analisi, esaminiamo le performance del nostro algoritmo a livello temporale. Per semplicità assumiamo l'utilizzo di una sola euristica per tutti gli esperimenti, ovvero quella basata sulla distanza. L'esperimento verrà eseguito su varie istanze con caratteristiche in comune e i risultati dati saranno mediati per calcolarne un andamento medio significativo. I primi casi analizzati, come presentati in Figura 5.1, trattano di grafi con 10 e 12



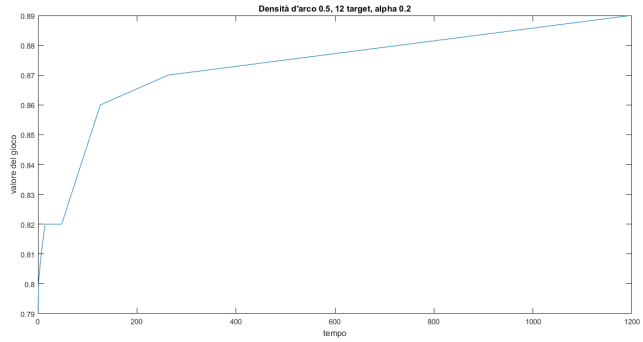
(a) *Grafo a 10 target, densità 0.25, α 0.2.*



(b) *Grafo a 12 target, densità 0.25, α 0.2.*



(c) *Grafo a 10 target, densità 0.5, α 0.2.*



(d) *Grafo a 12 target, densità 0.5, α 0.2.*

Figura 5.1: Andamento del valore del gioco in relazione al tempo, mediato su più istanze.

targets, con una densità d'arco che varia tra i 0,25 e i 0,5. Questa sperimentazione è volta a capire come l'algoritmo modifica le proprie performance al variare del numero di target, tenendo fissi gli altri parametri e al variare della densità d'arco. I dati su cui dobbiamo focalizzarci nell'analisi dei grafici sono:

- Il tempo che l'algoritmo impiega per raggiungere la soluzione ottimale;
- L'andamento della curva.

Nel caso in analisi, pare chiaro che il tempo per giungere alla soluzione ottimale dipende dal numero di target, che influisce enormemente sulla velocità dell'algoritmo, e dalla densità d'arco. Appare chiaro che in breve tempo viene determinata una soluzione sub-ottima, che viene poi migliorata con l'esecuzione prolungata dell'algoritmo. Come si può dunque dedurre, con un numero molto maggiore di target sarà molto difficile raggiungere la soluzione ottimale. Si possono tuttavia ottenere dei risultati comunque accettabili.

Per ottenere dei risultati migliori in minor tempo, si può ricorrere a un utilizzo di differenti euristiche per la selezione dei target da analizzare.

5.2.2 Confronto tra euristiche

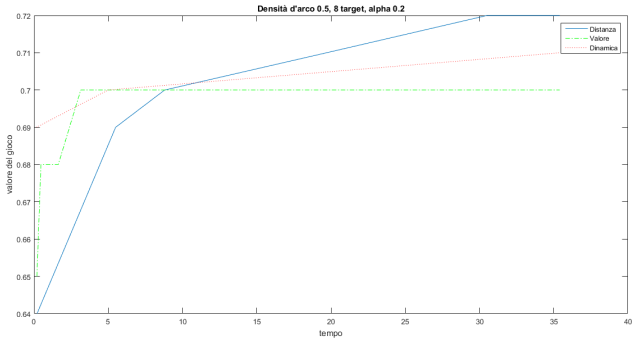
Come precedentemente discusso, abbiamo implementato tre euristiche: una basata sul valore dei target, una che li associa in base alla loro distanza e una che si aggiorna dinamicamente in base alle strategie dell'attaccante. Ci siamo focalizzati su creare degli esperimenti che testassero sulle medesime istanze tutte e tre le euristiche.

Le analisi effettuate si sono svolte su casistiche che variavano tra di loro un solo parametro, in modo da analizzare le variazioni delle performance in relazione al singolo parametro. Abbiamo infatti variato il numero di target, la densità d'arco e la probabilità di falso negativo.

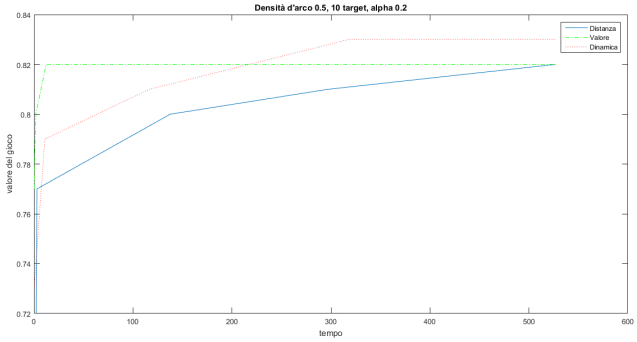
Gli esperimenti effettuati, come mostrato in Figura 5.2, mostrano in maniera chiara come nei vari casi le euristiche si comportano.

Nel caso base, presente in Figura 5.2(a), tutte le euristiche si comportano bene, con una differenza minima. L'euristica arriva velocemente a una buona soluzione iniziale, ovvero il best placement, siccome ha pochi target da analizzare, migliorandola successivamente. Anche con l'euristica basata sul valore si arriva velocemente a determinare un risultato ottimale.

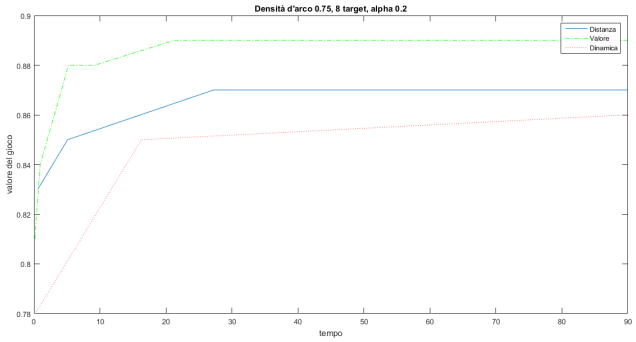
L'aumento del numero dei target, come mostrato in Figura 5.2(b), non influisce molto sulle performance dell'euristica basata sul valore, che ottiene il risultato migliore in breve tempo. Non vale la stessa cosa per le altre due, che con più target da analizzare richiedono più tempo. La dinamica



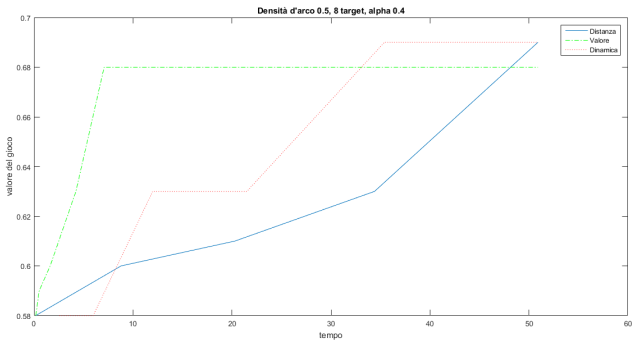
(a) *Grafo a 8 target, densità 0.5, α 0.2.*



(b) *Grafo a 10 target, densità 0.5, α 0.2.*



(c) *Grafo a 8 target, densità 0.75, α 0.2.*



(d) *Grafo a 8 target, densità 0.5, α 0.4.*

Figura 5.2: Confronto tra euristiche, con valore del gioco in relazione al tempo, mediato su più istanze.

necessita di tempo maggiore sia per il calcolo del best placement, sia per il continuo aggiornamento delle strategie dell'attaccante. L'euristica basata sulla distanza trova molti più vertici da analizzare.

L'aumento della densità d'arco, presente in Figura 5.2(c), anche in questo caso, non modifica la velocità con cui la risoluzione che utilizza l'euristica basata sul valore arriva alla soluzione. Cambia invece per la dinamica, che a causa della maggior connessione del grafo impiega più tempo a calcolare best placement e strategie dell'attaccante. Per quanto riguarda l'euristica sulla distanza, la connessione maggiore del grafo consente di analizzare velocemente cicli di copertura maggiori, ottenendo prima dei valori migliori.

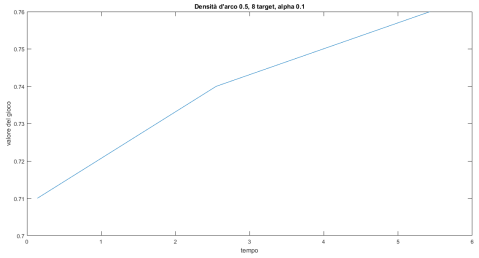
Per ultimo, l'aumento della probabilità di falso negativo mostrato in Figura 5.2(d). Esso influisce su tutte e tre le euristiche, in quanto diminuisce l'utilità data da cicli di copertura di minori dimensioni. Come si vede, l'andamento delle curve varia, mantenendo un valore di gioco basso nelle prime iterazioni, per poi crescere successivamente man mano che si trovano cicli di maggiori dimensioni.
Analizzando ora questo gruppo di esperimenti, pare chiaro che queste tre euristiche hanno utilizzi differenti in base alle casistiche in analisi.

- In caso di grafi con pochi target, l'euristica per valore e la dinamica sono le migliori;
- In casi con molti target, l'euristica per valore da risultati più velocemente. Anche la dinamica può essere utilizzata;
- In caso di grafi con bassa densità d'arco, l'euristica dinamica ha ottimo potenziale, come anche quella basata sul valore;
- In caso di elevata densità d'arco, l'euristica migliore è quella basata sulla distanza. Rimane sempre valida anche quella basata sul valore.

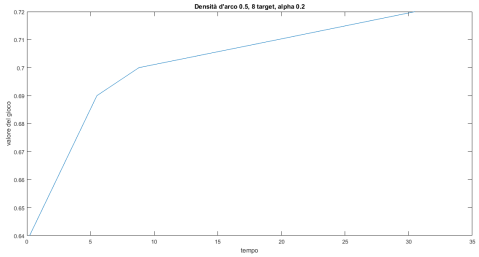
Vediamo ora più nel dettaglio come la probabilità di falso negativo influisce sulle prestazioni dell'algoritmo.

5.2.3 Affidabilità dei sensori

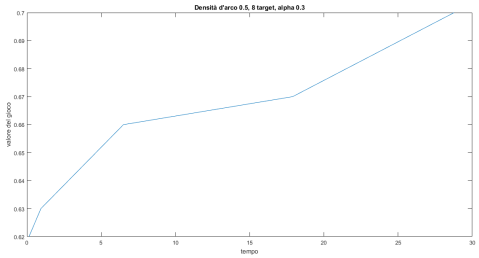
In questo esperimento analizziamo come il variare del parametro α modifica i grafici. Anche qui assumiamo di utilizzare unicamente l'euristica basata sulla distanza. Abbiamo omesso la casistica $\alpha = 0$, in cui la miglior strategia del difensore è data dal best placement, siccome i segnali sono totalmente affidabili e conviene posizionarsi nel punto migliore e limitarsi a rispondere ad eventuali allarmi.



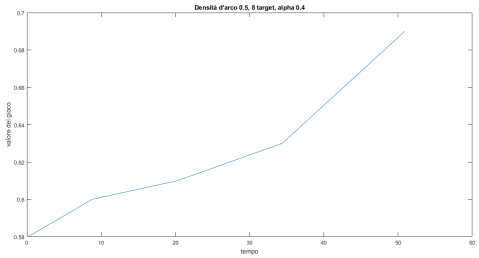
(a) *Grafo a 8 target, densità 0.5, α 0.1.*



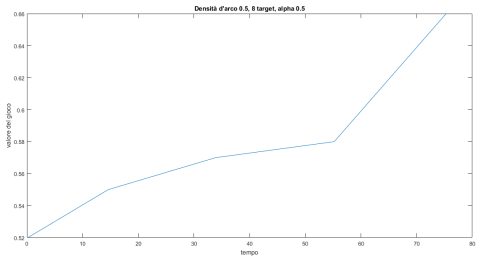
(b) *Grafo a 8 target, densità 0.5, α 0.2.*



(c) *Grafo a 8 target, densità 0.5, α 0.3.*



(d) *Grafo a 8 target, densità 0.5, α 0.4.*



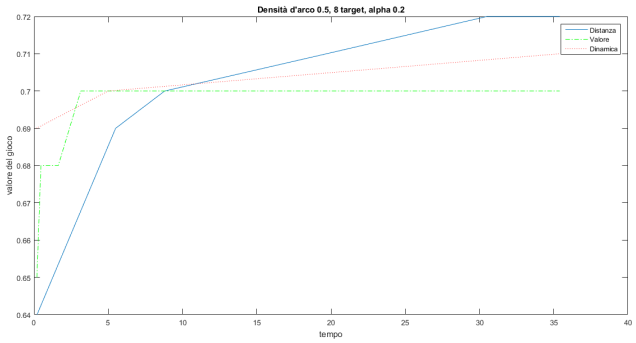
(e) *Grafo a 8 target, densità 0.5, α 0.5.*

Figura 5.3: Andamento del valore del gioco in relazione al tempo, mediato su più istanze, su vari valori di α.

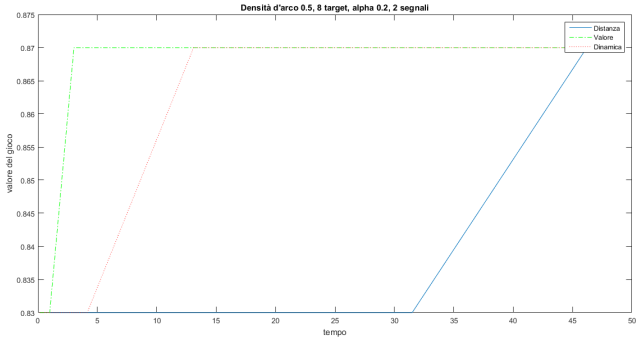
Come vediamo in Figura 5.3, maggiore è l'affidabilità dei segnali, migliore è la resa dell'algoritmo. Si può infatti notare che all'aumentare del parametro α la computazione richiede tempo maggiore per essere portata a termine e le soluzioni sub-ottime diventano sempre peggiori. Questo è dovuto al fatto che, coprendo meno target, l'attaccante avrà più probabilità di compiere con successo il suo attacco, grazie alla scarsa affidabilità dei sensori.

Questa analisi ci fornisce lo spunto per capire quanto l'idea di avere dei segnali il più affidabili possibili ci dia un grande aiuto, aumentando l'utilità attesa del difensore e migliorando le prestazioni del nostro algoritmo.

5.2.4 Quantità di segnali



(a) *Grafo a 8 target, densità 0.5, α 0.2, 1 segnale.*



(b) *Grafo a 8 target, densità 0.5, α 0.2, 2 segnali.*

Figura 5.4: Confronto in base al numero di segnali.

Analizziamo ora come il numero di segnali impiegati per proteggere i nostri obiettivi influisce sulle performance dell'algoritmo. Per questo confronto, come illustrato in Figura 5.4, abbiamo confrontato due casistiche,

una con un solo segnale che copre tutti i targets e l'altra in cui due segnali si dividono equamente gli obiettivi. Utilizziamo inoltre in entrambi i casi tutte e tre le euristiche a confronto, per poterne fare un ulteriore paragone sul numero di allarmi.

L'aumento dei segnali causa un leggero aumento del tempo necessario per calcolare il percorso ottimale del difensore. La differenza fondamentale è però l'utilità, che aumenta notevolmente. Questo è dovuto dal fatto che, siccome più segnali sono presenti, quando essi si attivano danno un miglior indirizzamento al difensore, che deve controllare solo un sottoinsieme dei possibili obiettivi.

Pertanto, se possibile, è buona norma inserire un numero maggiore di segnali, per poter incrementare l'utilità del pattugliatore. Con l'incremento dei segnali si nota inoltre che le euristiche migliori continuano a essere l'euristica basata sul valore e la dinamica, che portano al miglior cammino nel minor tempo.

5.2.5 Percorso randomico

Per completezza, abbiamo deciso di mostrare anche un confronto tra il metodo sviluppato in questa tesi e la creazione randomizzata del ciclo di copertura. Questo valore viene ottenuto risolvendo problemi di programmazione lineare, in modo da poter simulare un covering cycle potenzialmente casuale da parte del difensore.

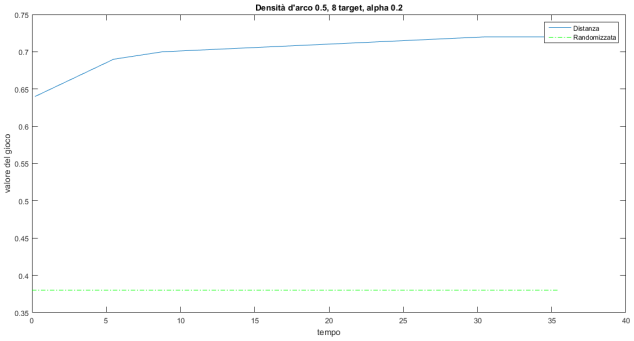
Le utilità di questo metodo, confrontate con quello esposto nella tesi (sempre utilizzando l'euristica basata sulle distanze) sono esposte in Figura 5.5.

Come si vede in maniera chiara, l'utilità del nostro metodo è sempre superiore. Appare chiaro, analizzando i vari grafici, che la randomizzata perde particolarmente in accuratezza all'aumentare della densità d'arco dei grafi. Per ottenere valori accettabili occorrerebbe utilizzare grafi poco connessi e con pochi vertici, cosi da poterci avvicinare a risultati più soddisfacenti.

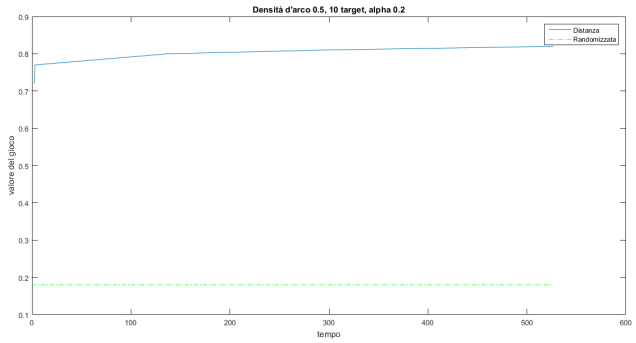
5.3 Istanze realistiche

Analizziamo ora anche dei casi reali, in modo da avere una più completa visione delle performance del nostro algoritmo. Per fare questa analisi, utilizziamo la mappa del museo del Louvre. Per semplicità verranno eseguiti i calcoli solo sull'istanza relativa al secondo piano del museo.

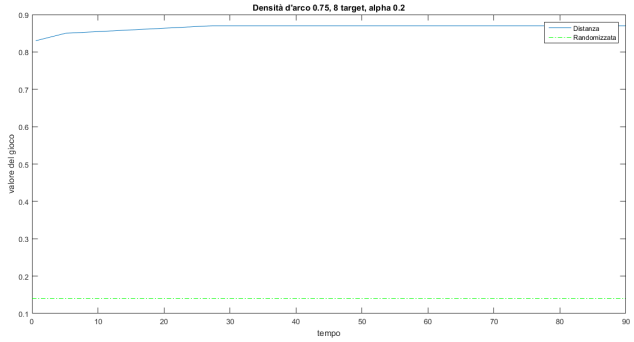
Partendo dalla mappa del piano, costruiamo il grafo utilizzando le stanze come vertici e i corridoi come archi. A questi nodi assegnamo un valore π



(a) *Grafo a 8 target, densità 0.5, α 0.2.*



(b) *Grafo a 10 target, densità 0.5, α 0.2.*



(c) *Grafo a 8 target, densità 0.75, α 0.2.*

Figura 5.5: Confronto con metodo di selezione randomico.

calcolato in relazione all'importanza delle opere in esso contenute, pesato in base alle dimensioni della stanza. Assegnamo inoltre una deadline d, calcolata in base al tempo di fuga necessario al malintenzionato per fuggire, stimato con la distanza del vertice dalla più vicina via di fuga. Agli archi è assegnato un peso pari alla distanza effettiva dei nodi che essi collegano. Mostriamo in Figura 5.6 il grafo relativo all'istanza analizzata.

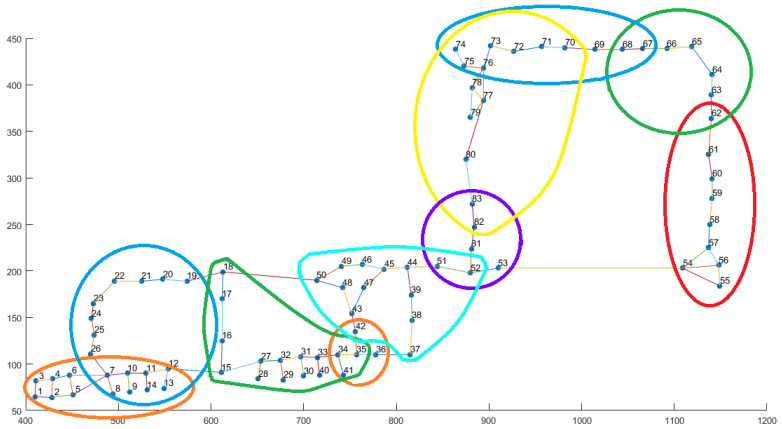


Figura 5.6: Grafo ottenuto tramite la mappa del secondo piano del Louvre.

Abbiamo evidenziato su tale grafo una possibile copertura dei nodi da parte di 11 segnali. Essi offrono un'incertezza spaziale, coprendo ciascuno più target. Tuttavia, inseriamo la possibilità che un vertice possa essere coperto da più allarmi contemporaneamente.

Con tale configurazione eseguiamo tutte e tre le euristiche, per un tempo massimo di 1 ora, in modo da analizzare i risultati ottenuti in questo periodo. Il grafico delle performance di tale algoritmo è presentato in Figura 5.7.

Vista la dimensione del grafo, in un'ora non si riesce a ottenere una soluzione ragionevole. Tuttavia, analizzando quanto siamo riusciti a calcolare, si vede chiaramente che l'euristica basata sul valore è sempre la più performante, seguita dall'euristica dinamica. Come potevamo ipotizzare, data la bassa densità d'arco del grafo utilizzato, l'euristica basata sulla distanza è la più lenta.

Non possiamo tuttavia effettuare un'analisi completa, in quanto questi sono i dati ottenuti in un'ora di esecuzione. Effettuando le stesse computazioni per maggior tempo e su macchine con prestazioni più elevate potremmo ipoteticamente analizzare grafici più accurati per mostrare in modo migliore l'andamento delle tre euristiche.

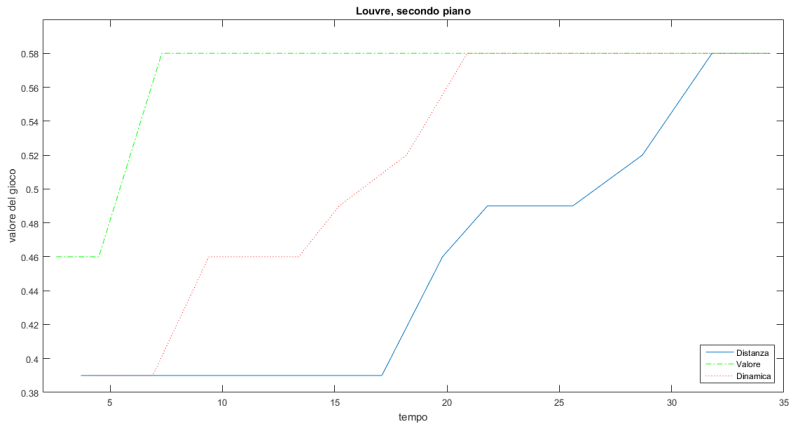


Figura 5.7: Confronto tra euristiche sul secondo piano del Louvre.

Capitolo 6

Conclusione e lavori futuri

"Possiamo vedere solo poco davanti a noi,
ma possiamo vedere tante cose che bisogna fare."

Alan Turing

Tiriamo ora le conclusioni di questo lavoro di ricerca, analizzando poi tutte le possibili continuazioni di questo lavoro per il futuro.

6.1 Conclusioni

In questa ricerca tutto è iniziato con lo scopo di proteggere degli obiettivi sensibili da dei malintenzionati. Abbiamo dapprima analizzato i *Patrolling Security Games*, focalizzandoci su sistemi protetti da allarmi spazialmente incerti. Come estensione per questa tesi è stato analizzato il caso in cui questi allarmi non siano infallibili, ma soggetti a possibili errori di rilevazione detti falsi negativi. Abbiamo dunque scoperto che in questi casi la miglior azione possibile per un difensore è pattugliare per proteggere i targets, in quanto non può affidarsi totalmente ai segnali.

Abbiamo inoltre analizzato tre euristiche per l'enumerazione dei targets da pattugliare, in modo da ordinarli sotto vari criteri. Abbiamo poi fatto vari esperimenti su casi con un vario numero di obiettivi da proteggere, variando la densità d'arco e modificando la probabilità di falso negativo dei sensori.

A questo punto, abbiamo scoperto come l'algoritmo analizzato scala al variare di questi parametri e quali euristiche sono migliori a seconda dei casi in esame.

6.2 Sviluppi futuri

Terminato questo lavoro, vari scenari si presentano per il futuro. Molte ricerche sono ancora da effettuare in questo ambito, per poterlo migliorare e per poter gestire tutte le possibili casistiche.
I più importanti lavori ed estensioni che si potrebbero effettuare sono:

- *Aumentare le unità preposte alla difesa.* Come succede spesso in casi reali, a difendere un'area in cui sono collocati degli obiettivi sensibili sono più pattuglie, che seguono rotte di pattugliamento distinte. Diventa pertanto possibile gestire questa casistica, aggiungendo al caso base un numero maggiore di difensori che possono avere strategie e rotte differenti tra loro.

- *Come gestire il posizionamento dei segnali.* Nelle nostre istanze di prova i segnali sono stati collocati coprendo dei target randomici. Potrebbe essere un'estensione molto interessante se si potesse calcolare il miglior posizionamento dei segnali per garantire un'utilità maggiore.

- *Introduzione di falsi positivi.* Abbiamo gestito i falsi negativi con questo lavoro, tuttavia esistono casi in cui il segnale si attiva senza che ci sia un tentativo di furto in atto. Diventa necessario perciò introdurre anche questo tipo di imprecisione degli allarmi, che rende il modello più realistico.

- *Ottimizzazione degli algoritmi.* A livello algoritmico, è sempre necessario un continuo lavoro di ottimizzazione, in modo da rendere il modello capace di gestire casi di maggiori dimensioni e di ridurre il tempo necessario alle computazioni. Questo ci consentirebbe di analizzare anche casi reali complessi. Essi sono infatti caratterizzati, come abbiamo visto, da un elevato numero di target e segnali, con una densità d'arco solitamente bassa. Dato ciò, euristica per valore e dinamica si prestano bene a questa analisi, ma potrebbero terminare l'esecuzione dopo una grande quantità di tempo. Rimane chiaramente la possibilità di ottenere in qualsiasi momento una soluzione sub-ottima, ma idealmente tenderemmo a preferire un'analisi che porti alla soluzione ottima.

Bibliografia

[1] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184-185:78–123, 2012.

[2] Nicola Basilico, Nicola Gatti, and Federico Villa. Asynchronous multi-robot patrolling against intrusion in arbitrary topologies. In *AAAI*, pages 1224–1229, 2010.

[3] Eddie Dekel and Faruk Gul. Rationality and knowledge in game theory. *Econometric Society Monographs*, 26:87–172, 1997.

[4] S. Kraus E. Sless, N. Agmon. Multi-robot adversarial patrolling: facing coordinated attacks. In *Autonomous Agents and Multi-Agent Systems(AAMAS)*, pages 1093–1100, 2014.

[5] David Gale. A theory of n-person games with perfect information. *Proceedings of the National Academy of Sciences of the United States of America*, 39(6):496, 1953.

[6] Manish Jain, Bo An, and Milind Tambe. An overview of recent application trends at the AAMAS conference: Security, sustainability, and safety. *AI Magazine*, 33(3):14–28, 2012.

[7] Harold W Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, 2(28):193–216, 1953.

[8] Roberto Lucchetti. *A Primer in Game Theory.* Società Editrice Esculapio, 2011.

[9] Y. Emaliah P. Stone C. Julien S. Vishwanath N. Agmon, C. Fok. On coordination in practical multi-robot patrol. In *IEEE International Conference on Robotics and Automation(ICRA)*, pages 650–656, 2012.

[10] John F. Nash. Non-cooperative games. *Annals of Mathematics*, 54:286–295, 1951.

[11] Giuseppe De Nittis Nicola Basilico and Nicola Gatti. Adversarial patrolling with spatially uncertain alarm signals. *arXiv preprint arXiv:1506.02850*, 2015.

[12] Martin J. Osborne. *An Introduction to Game Theory*. OUP USA, 2004.

[13] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: The application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, volume 1, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[14] Jason Tsai, Shyamsunder Rathi, Fernando Ordóñez, Christopher Kiekintveld, and Milind Tambe. Iris: a tool for strategic security allocation in trasportation networks. *AAMAS*, pages 37–44, 2009.

[15] Albert W. Tucker. On jargon: The prisoner's dilemma. *UMAP Journal*, 1:101, 1980.

[16] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[17] John von Neumann and Oskar Morgenstern. Theory of games and economic behavior. *Bullettin of American Mathematical Society*, 51:498–504, 1945.

[18] Heinrich F. von Stackelberg. *Marktform und Gleichgewicht*, volume 1. Springer, 1934.