Capitolo 3

GA per l'Allocazione di Task

I recenti conflitti militari hanno dimostrato il valore strategico degli UAVs. Il ruolo degli UAVs è passato da semplici missioni di ricognizione a missioni di carattere offensivo. Le capacità degli UAVs migliorano notevolmente se più UAVs cooperano tra loro. La realizzazione della cooperazione richiede un metodo di assegnamento di task (e.g. ricerca/classificazione, attacco e valutazione dei danni da battaglia su potenziali obbiettivi) ai velivoli in modo che tutti gli obbiettivi siano raggiunti e i vincoli siano rispettati. Il problema di allocazione dei task può essere visto sia

staticamente che dinamicamente. L'allocazione di task statica comporta che l'assegnamento, fatto ad un tempo t, sia fisso, mentre l'assegnamento dinamico è fatto ad ogni intervallo di tempo prefissato oppure in presenza di un dato evento.

Esistono in letteratura una grande varietà di metodi per risolvere il problema. Approssimativamente, possono essere classificati nelle seguenti categorie: ottimizzazione di una rete di flusso, programmazione lineare intera, e approccio fuzzy. Data la natura "intrattabile" del problema d'allocazione di task, è infatti un problema combinatorio Non Polinomiale Completo che in pratica è irrisolvibile con un algoritmo lineare standard se la complessità del problema è abbastanza elevata, e la sua importanza nel controllo cooperativo è necessario esplorare altre strade per sviluppare dei buoni algoritmi euristici per il problema.

In questo lavoro il problema è stato formulato in modo da minimizzare il tempo di completamento della missione, anche sotto vincoli temporali sui velivoli, e da minimizzare il rischio di fallimento della missione in base a delle possibili capacità operative dei velivoli e minacce dei target. L'algoritmo di allocazione usato si basa sui principi degli algoritmi genetici.

3.1 Formulazione del Problema

Consideriamo uno scenario in cui un gruppo di M UAVs (agenti) devono eseguire N task, ossia attraversare N obbiettivi, e quindi raggiungere una predefinita posizione (*rendez-vous*). Il problema cosi posto risulta un MTSP (*Multiple Traveling Salesperson Problem*).

Nella regione di interesse, sono presenti un numero di obiettivi, alcuni conosciuti a priori, altri che vengono individuati quando un UAV si trova nelle loro vicinanze (solo in simulazione, nel caso dinamico), "pop-up"; sono, inoltre, presenti degli ostacoli statici.

L'ambiente di lavoro può essere scomposto in più sottoproblemi:

1. In primo luogo, dati M agenti con N target, assegnare ogni agente ad un target in modo che siano soddisfatti gli obbiettivi voluti.
2. Secondo, per ogni agente assegnato ad uno specifico obbiettivo individuare un percorso (attraverso specifici waypoint) che consenta all'agente di raggiungere l'obbiettivo assegnatoli evitando gli ostacoli presenti nella regione di interesse.

3.1.1 MTSP

Il MTSP è una generalizzazione del classico TSP, in cui si tiene conto della necessità di considerare la presenza di più "commessi viaggiatori". Come nel TSP classico, in cui si cerca di trovare il percorso più economico, cioè un percorso chiuso che passa attraverso ciascun nodo del grafo (città) e il cui costo totale in termini di distanza percorsa sia minimo, il MTSP minimizza i percorsi di più viaggiatori in modo che tutti i nodi di un grafo siano visitati con il vincolo che una città sia visitata da un unico viaggiatore.

3.1.2 Pianificazione del percorso

Il problema fondamentale della pianificazione di percorso si può esprimere in forma semplificata come segue:

- sia A un singolo corpo rigido - il velivolo - che si muove in uno spazio euclideo W, detto *spazio di lavoro* (*workspace*), rappresentato da R^n , dove n=2,3;
- siano B_1,.....,B_q corpi rigidi fissi distribuiti in W, che rappresentano degli ostacoli;

- si assuma che sia la geometria di $A,B_1,.....,B_q$ sia la posizione degli ostacoli in W siano note in modo accurato;

- il problema è il seguente: date la posizioni ed orientazioni iniziali e finali di A in W, generare un percorso, specificando una sequenza continua di posizioni ed orientazioni di A evitando il contatto con gli ostacoli, partendo dalla posizione ed orientazione iniziali e terminando alla posizione ed orientazione finali.

3.1.2.1 Modellazione del workspace

La regione di ricerca è assunta quadrata con lunghezza L. Nella regione di interesse gli agenti e gli obbiettivi sono visti come punti; gli ostacoli sono modellati come poligoni, e vengono individuati dai loro vertici.

Come si vede in Figura, la regione di ricerca è normalizzata rispetto alla lunghezza L. Inoltre, le velocità dei velivoli, che considereremo per semplicità costanti, sono uguali; e vengono normalizzate anch'esse rispetto alla lunghezza L della regione di ricerca. In questo modo i tempi di percorrenza di un generico percorso nella regione reale e nella regione normalizzata sono uguali.

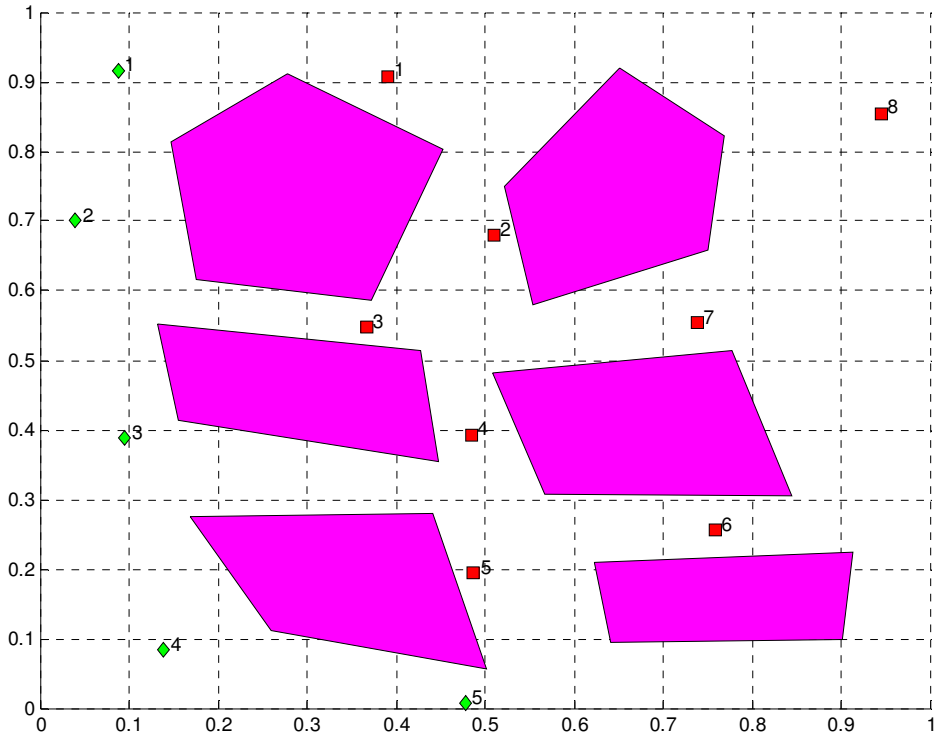

Figure1. Possibile Regione di Ricerca

I velivoli sono rappresentati dai triangoli verdi; gli obiettivi dai rettangoli rossi; in ogni scenario il target con indice maggiore rappresenta il punto di rendez-vous.

3.1.2.2 Grafo di Visibilità

Il metodo dei *grafi di visibilità* è quello di costruire un percorso rappresentato da una spezzata che unisce il punto di partenza con quello di arrivo passando per i vertici degli ostacoli. Vale la seguente:

Definizione: un *grafo di visibilità G* è specificato dalle seguenti proprietà:

- i nodi del grafo sono rappresentati dalle posizioni dello start, e del goal e dai vertici degli ostacoli presenti nello spazio di lavoro

- due nodi del grafo sono collegati fra loro se e solo se esiste un segmento che li unisce che appartenga alla frontiera degli ostacoli o che appartenga interamente allo spazio libero, ad eccezione dei due estremi.

In pratica per disegnare il grafo di visibilità si considerano tanti nodi quanti sono i vertici dei poligoni che rappresentano gli ostacoli, si aggiungono quindi altri due nodi corrispondenti alle posizioni iniziale e finale, e si collegano due nodi solo quando essi "si vedono", nel senso che è possibile individuare una linea retta che li unisce senza intersecare gli ostacoli (vedi figura).

Figure 2. Grafo di Visibilità

Un semplice algoritmo per costruire il grafo G consiste nel considerare ogni coppia di punti (X,X'), dove X ed X' possono essere sia la posizione del velivolo, o dell'obbiettivo, sia vertici degli ostacoli; se X ed X' sono gli estremi di uno stesso lato di un ostacolo, allora si collegano i nodi corrispondenti di G. In caso contrario si calcola l'intersezione fra gli ostacoli e la retta passante per X ed X': si possono unire i nodi corrispondenti del grafo se e solo se non si riscontra alcuna intersezione all'interno del segmento aperto passante per i punti; questo algoritmo richiede un tempo di calcolo di ordine $O(n^3)$,dove n è il numero totale di vertici degli ostacoli.

Una volta costruito il grafo, si deve utilizzare un algoritmo che ricerchi un possibile percorso che unisca il velivolo e il target; in tal caso il tempo richiesto per il calcolo è proporzionale al numero l di connessioni fra i nodi del grafo O(l). Se si desidera invece il percorso più breve si deve utilizzare un algoritmo quadratico, funzione del numero di nodi $O(n^2)$.

Per costruire il grafo di visibilità si è finora supposto che gli ostacoli siano poligonali; si può generalizzare quanto detto in precedenza al caso più realistico di ostacoli di forma generica. In questo caso, non preso in considerazione in questo lavoro, bisogna introdurre le definizioni di *Poligono Generalizzato* e *Grafo di Visibilità Generalizzato.*

Per concludere si può osservare che i grafi di visibilità hanno la proprietà di individuare il più breve fra i percorsi possibili, in termini di norma euclidea; per

contro, la complessità degli algoritmi di generazione del grafo e di ricerca del percorso cresce più che linearmente all'aumentare del numero dei nodi; inoltre il percorso generato si svolge ancora ad una distanza troppo ridotta dagli ostacoli.

3.1.2.3 Ricerca dei Percorsi

L'algoritmo utilizzato per la ricerca del percorso più breve, a partire dal Grafo di Visibilità, è l'algoritmo di Dijkstra. Per usare l'algoritmo di Dijkstra il grafo viene ridisegnato, in modo da trasformare il problema dato in uno su *grafo aciclico*.

Definizione: Si definisce *grafo aciclico* un grafo in cui non sono presenti cicli, ossia dati un nodo iniziale n_1 e un nodo finale n_r, non esiste un percorso per cui $n_1=n_r$ con $n_h \neq n_k$ nodi intermedi.

Su tale nuovo grafo viene usato un procedimento di etichettatura, cioè un metodo che associa a ogni nodo un valore che rappresenta il costo per andare dal nodo di origine al nodo etichettato.

Metodo di Dijkstra

1. Etichetta, con il costo del cammino diretto dal nodo 1, tutti gli altri nodi (questa costituisce la prima etichettatura).

2. Segna con un * il nodo con l'etichetta minore: per tale nodo è stata già trovata la soluzione ottima e perciò nei passi successivi esso non viene considerato.
3. Se esistono ancora nodi senza * vai al passo 4, se no stop.
4. Per tutti i nodi non segnati con *, calcola il minimo tra l'etichetta precedente e il costo del cammino proveniente dall'ultimo nodo contrassegnato con *, questa è la nuova etichetta. Vai al passo 2.

Per calcolare il numero di operazioni elementari da eseguire e la loro molteplicità ricordiamo che abbiamo chiamato etichettatura l'insieme delle etichette di una iterazione e che a ogni iterazione diminuisce di una unità il numero dei nodi da considerare.

Sono necessarie pertanto n-1 etichettature.

- Per l'etichettatura i-esima si devono considerare n-i nodi.
- Per etichettare un nodo si devono confrontare 2 cammini.
- Per ogni confronto si deve fare una somma.

I nodi da etichettare sono quindi (n-1) + (n-2) + … + 2 + 1 e le operazioni necessarie per etichettare un nodo sono 3. Il numero totale di operazioni elementari necessarie è allora

$3 \times n/2 \times (n-1) = 3/2 \times (n^2-1) = O(n^2)$.

Poiché la complessità è polinomiale, l'algoritmo di Dijkstra si può dire efficiente: ha una complessità computazionale che cresce con il quadrato del numero dei nodi.

3.1.3 Descrizione del GA

Tipicamente, il GA è caratterizzato dai seguenti componenti:

- Una rappresentazione genetica (*codifica*) di una possibile soluzione del problema
- Una popolazione di soluzioni
- Una funzione di Fitness che valuta la "bontà" di ogni soluzione
- Operatori genetici che generano una nuova popolazione a partire dalla popolazione esistente

3.1.3.1 Codifica del Cromosoma

La missione del team UAVs è rappresentata da una "stringa" di valori, con codifica intera, allo scopo di modellare l'interazione tra gli agenti che cooperano per raggiungere sia gli obbiettivi individuali che quelli del team stesso.

L'utilizzo della codifica intera permette di considerare il singolo individuo (il cromosoma) come un vettore di dimensione M (numero dei target da cui escludiamo il target che rappresenta il punto di rendez-vous), in cui ogni elemento rappresenta un possibile target; la posizione all'interno del vettore indica la sequenza di visita dei vari obbiettivi. Al vettore dei target viene associato un ulteriore vettore di dimensione N (numero di agenti) che indica il numero di target assegnato ad ogni velivolo; per un dato velivolo l'assegnamento può variare da zero a M, ciò corrisponde ad una situazione in cui un unico individuo del team copre tutti i targets mentre gli altri procedono direttamente al rendez-vous.

Considerando lo scenario di Figura 1 un possibile individuo è il seguente:

[2 3 5 1 6 4 8 7] *sequenza di target*

[0 2 1 5 0] *numero di target per agente*

Questo comporta i seguenti assegnamenti:

V1: va direttamente al rendez-vous (target 8 in Figura 1)
V2: passa da 2, va su 3 e infine al rendez-vous
… e cosi via.

3.1.3.2 Popolazione Iniziale

Il secondo passo nella realizzazione di un GA è quello di generare una popolazione di possibili allocazioni. La soluzione migliore è quella di utilizzare alcuni individui che

codificano allocazioni note a priori(da rivedere), il resto della popolazione può essere ottenuta generando dei cromosomi casualmente. Questo assicura un buon punto di partenza che comporta una rapida convergenza, sebbene è possibile partire da una popolazione completamente casuale.

3.1.3.3 Funzione Fitness

Le funzioni di Fitness, che verranno analizzate nel prossimo Capitolo, sono state costruite in modo da risolvere le missioni analizzate. Partendo da una funzione di costo da minimizzare, si mappa il suo valore in un valore di fitness. Tale valore rappresenta un premio basato sulle performance della possibile soluzione; migliore è la soluzione, più alto è il valore di fitness. Minimizzare la funzione di costo significa assegnare, quindi, un valore alto alla funzione di Fitness. Poiché il GA cerca di generare individui "buoni" per migliorare il Fitness, una migliore allocazione di target è ottenuta attraverso una migliore fitness nel GA.

3.1.3.4 Operatori Genetici

Gli operatori genetici utilizzati sono tutti degli operatori di mutazione. Poiché il generico individuo è composto da due vettori, sono state realizzate funzioni che permettono di modificare sia la sequenza dei target, sia il numero di velivoli assegnati

ad ogni agente. Sostanzialmente le funzioni realizzate si basano sulle seguenti funzioni:

- Funzioni che scambiano le posizioni di elementi dei due vettori
- Funzioni che permutano o traslano un numero casuale di elementi