Pianificazione Ottima del Moto

Dott. Lucia Pallottino, Ing. Adriano Fagiolini

October 2005

Indice

Capitolo 1

Introduzione

Questa parte del corso di Robotica riguarda il problema della pianificazione ottima delle traiettorie. Queste dispense si basano in gran parte sul testo del Prof. Steve M. LaValle "Planning Algorithm" disponibile in rete. Dopo una introduzione al problema verrano illustrati alcuni tra i metodi risolutivi classici noti in letteratura. Si inizierà con il problema della pianificazione su spazi discreti nel capitolo 2 con i relativi metodi risulutivi. Nel capitolo 3 si analizzerà invece il problema della pianificazione ottima nel caso si spazi continui e si mostreranno alcuni casi in cui i metodi risolutivi descritti consentono di determinare una soluzione del problema in forma analitica.

Capitolo 2

Pianificazione discreta

In questo capitolo vengono descritti alcuni metodi per la ricerca di traiettorie ammissibili su spazi discreti. Il problema della pianificazione delle traiettorie viene formalizzato come problema di ricerca di cammini ammissibili su un grafo. Si illustreranno dapprima diversi metodi di ricerca di soluzioni ammissibili, in cui si è interessati a raggiungere la configurazione desiderata. Successivamente verranno descritti metodi per la ricerca di soluzioni ottime.

2.1 Formalizzazione del problema

Sia $\mathcal{X}$ lo *spazio di stato* discreto, cioè numerabile o anche finito. Sia $\mathcal{U}$ lo *spazio dei controlli* ammissibili che in generale può dipendere dallo stato particolare in cui si trova il sistema. In tal caso lo spazio dei controlli associato ad uno stato viene indicato con $\mathcal{U}(x)$ mentre lo spazio dei controlli risulta $U = \cup_{x \in \mathcal{X}} \mathcal{U}(x)$. Sia $f : \mathcal{X} \times \mathcal{U}(x) \to \mathcal{X}$ la *funzione di transizione degli stati*, i.e. ogni controllo u applicato ad uno stato x produce un nuovo stato $x' = f(x, u)$. Sia infine $\mathcal{X}_G \subset \mathcal{X}$ lo *spazio degli stati di arrivo*.

Lo scopo degli algoritmi di pianificazione è quindi quello di determinare una sequenza finita di controlli che trasformano uno stato iniziale x_I in uno stato in $\mathcal{X}_G$.

Spesso è conveniente modellizzare il problema della pianificazione ammissibile come il problema della ricerca di cammini su un grafo orientato. Si considerino infatti gli stati in $\mathcal{X}$ come nodi di un grafo, un arco tra gli stati x e x' esiste nel grafo se e soltanto se esiste $u \in \mathcal{U}(x)$ tale che $x' = f(x, u)$. Lo stato di partenza e gli stati di arrivo sono dei particolari nodi del grafo. Il problema della pianificazione ammissibile diventa quindi il problema di determinare se esiste un cammino sul grafo a partire dal nodo iniziale sino ai nodi finali.

2.1.1 Esempio di pianificazione discreta: il labirinto

Si consideri il labirinto riportato in figura 2.1 per il quale si vuole trovare una traiettoria dallo stato iniziale x_I sino allo stato finale x_{G_1} o allo stato finale x_{G_2}. Si procede col numerare tutti gli stati del problema come riportato nella figura 2.2. Ogni stato del sistema viene rappresentato, come riportato in figura 2.3, come un nodo del grafo. Infine si determinano gli archi che appartengono al grafo. Per l'esempio considerato, il grafo associato al problema della pianificazione discreta ammissibile è riportato in figura 2.4. Si noti che non esiste una traiettoria ammissibile che porti lo stato x_I nello

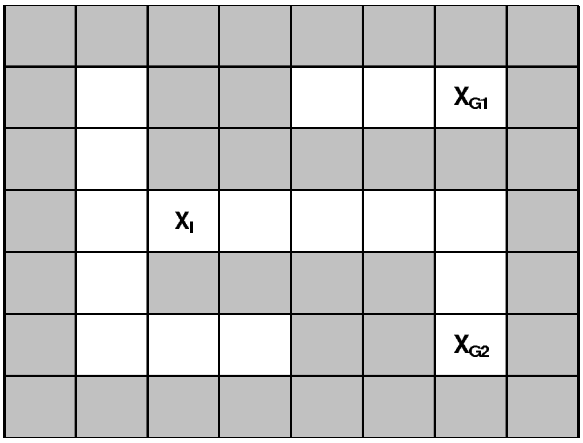


Figura 2.1: Esempio di pianificazione discreta

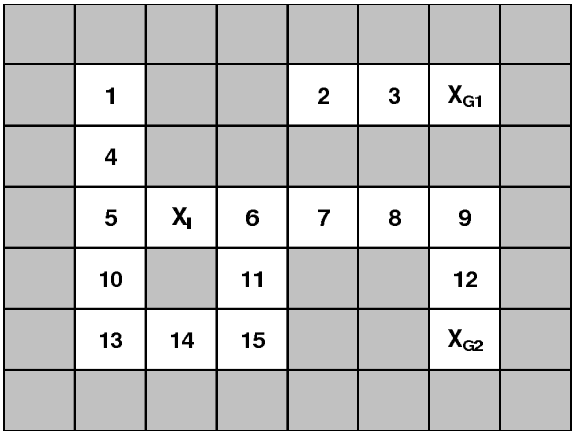


Figura 2.2: Numerazione degli stati discreti

stato x_{G_1}, così come non esiste un cammino sul grafo che connetta i rispettivi nodi. Lo stato x_{G_2} è invece raggiungibile tramite due diverse traiettorie, così come esistono due cammini distinti sul grafo che connettono il nodo x_I con il nodo x_{G_2}.

2.2 Ricerca di traiettorie ammissibili

Presentiamo ora alcuni metodi di ricerca su grafi per trovare la soluzione al problema delle traiettorie ammissibili. Ogni algoritmo di ricerca sul grafo deve essere *sistematico*. Se il grafo è di dimensione finita questo vuol dire che l'algoritmo deve visitare ogni stato raggiungibile e quindi deve essere in grado di determinare in tempo finito se esiste o meno una soluzione. Per essere sistematico l'algoritmo deve tenere traccia degli stati già visitati in modo tale da evitare di visistare ciclicamente gli stessi stati.

Nel caso di grafi infiniti si richiede agli algoritmi di ricerca che se la soluzione esista questa venga trovata in tempo finito altrimenti si accetta che l'algoritmo possa procedere per un tempo infinito.

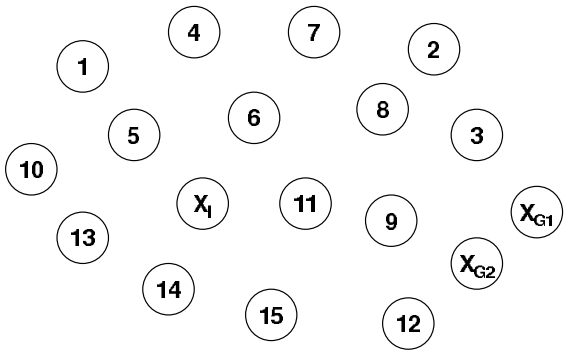


Figura 2.3: Ad ogni stato discreto si associa un nodo del grafo

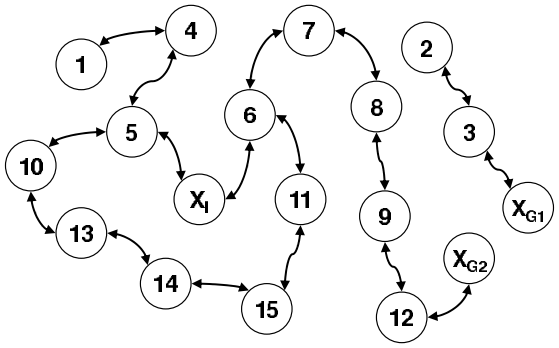


Figura 2.4: Si determinano quali sono gli archi presenti nel grafo

2.2.1 Metodi di ricerca in avanti

Lo schema generale per gli algoritmi di ricerca risulta

Ricerca in avanti

```
1  Q.Insert(x_I)
2  while Q not empty do
3        x ←Q.GetFirst()
4        if  x ∈ X_G
5            return SUCCESS
6        forall u ∈ U(x)
7            x' ← f(x, u)
8            if x' not visited
9                Mark x' as visited
10               Q.Insert(x')
11           else
12               Resolve duplicate x'
13 return FAILURE
```

Durante la ricerca nel grafo ci sono tre diversi tipi di stato:

Non visitato: Stato che non è ancora stato visitato. Inizialmente ogni stato, con l'esclusione di x_I, è di questo tipo.

Morto: Stato che è stato visitato e per il quale ogni possibile stato successivo è stato visitato. Lo stato successivo di x è uno stato x' per cui esiste un ingresso $u \in \mathcal{U}(x)$ tale che $x' = f(x, u)$.

Vivo: Stato che è già stati visitati ma ancora non tutti gli stati successivi lo sono stati. Inizialmente l'unico stato vivo è x_I.

L'insieme degli stati vivi viene ordinato secondo una coda di priorità Q. L'unica differenza sostanziale tra i vari algoritmi di ricerca è la funzione utilizzata per ordinare la coda Q. La coda più utilizzata è quella FIFO (First-In First-Out) per la quale lo stato che è stato inserito per primo nella coda è anche il primo ad essere scelto quando Q.GetFirst() viene chiamata.

Tale algoritmo è soltanto in grado di determinare se una soluzione esiste o meno. Se si vuole anche ottenere una traiettoria ammissibile è necessario associare allo stato x' lo stato genitore x.

Alcuni algoritmi richiedono di calcolare un certo costo e associarlo ad ogni stato. Se uno stesso stato viene raggiunto più volte il costo potrebbe essere modificato. Questo costo può essere utilizzato per ordinare la coda Q oppure per consentire la ricostruzione della traiettoria una volta che l'algoritmo è completato. Un algoritmo di questo tipo è l'algoritmo di Dijkstra descritto nel prossimo paragrafo.

2.2.1.1 Algoritmo di Dijkstra

Un algoritmo che consente di trovare sia le traiettorie che le traiettorie ottime è l'algoritmo di Dijkstra. Questo algoritmo consente la ricerca di cammini minimi su grafi a partire da un nodo sorgente ed è una particolare forma di programmazione dinamica.

Si supponga che, nella rappresentazione a grafo del problema della pianificazione del moto, ad ogni arco $e \in E$ sia associato un costo $l(e)$ non negativo. Tale costo rappresenta il costo per applicare l'azione u che porta uno stato x in $x' = f(x, u)$. Per questo motivo il costo $l(e)$ si rappresenta anche come $l(x, u)$. Il costo totale del cammino sul grafo risulta quindi pari alla somma dei costi sugli archi che a partire dal nodo iniziale portano a quello finale.

La coda di priorità Q è ordinata secondo una funzione $C : \mathcal{X} \to [0, \infty]$ detta *cost-to-come*. Per ogni stato x, $C^*(x)$ viene detta cost-to-come ottima di cammini che partono dallo stato iniziale x_I e arrivano sino a x. Ovviamente, il costo ottimo si ottiene considerando il minimo tra i costi di tutti i cammini che connettono x_I con x.

Il cost-to-come viene calcolato incrementalmente durante l'esecuzione dell'algoritmo di ricerca. Inizialmente si ha che $C^*(x_I) = 0$. Sia $x' = f(x, u)$ allora $C(x') = C^*(x) + l(e)$, dove e è l'arco che connette il nodo x con il nodo x'. In questo caso $C(x')$ rappresenta il cost-to-come conosciuto per x' ma non sappiamo ancora se sia il valore ottimo. Quello che può succedere è infatti che, come previsto dalla riga 12 dell'algoritmo generale in 2.2.1, il nodo x' è già in Q e viene raggiunto attraverso un nuovo cammino a costo minore. In tal caso il costo $C(x')$ viene aggiornato e Q viene riordinato di conseguenza.

Una volta che il nodo x' viene restituito dalla funzione Q.GetFirst() allora diventa un nodo morto e per induzione è possibile far vedere che non è possibile raggiungere il nodo a costo minore del corrente valore $C(x')$. Infatti, per induzione, il caso base è dato da $C^*(x_I) = 0$ e supponiamo che tutti i nodi morti hanno associato il proprio cost-to-come ottimo che non può più essere modificato. Sia x il nodo

restituito dalla funzione Q.GetFirst(), il valore $C(x)$ è ottimo perchè ogni altro cammino con costo minore deve passare prima attraverso ad un altro nodo in Q che però ha sicuramente costo maggiore di x. Tutti i cammini che passano soltanto attraverso stati morti sono stati già considerati per produrre $C(x)$. Quindi se il nodo x è morto $C(x) = C^*(x)$.

Si consideri ora il grafo riportato in figura 2.5. Inizialmente $Q = \{x_I\}$ e $C^*(x_I) = 0$. I nodi

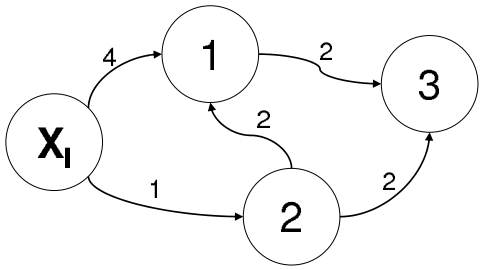


Figura 2.5: Determinare i cammini minimi a partire da X_I.

successori di x_I sono 1 e 2 che hanno associato un cost-to-come pari a $C(1) = 0 + 4 = 4$ e $C(2) = 0 + 1 = 1$, da cui $Q = \{2, 1\}$. Il nodo da esaminare è quindi 2 i cui successori sono 1 e 3 mentre il costo del nodo 3 risulta $C(3) = C(2) + 2 = 3$ il costo di 1 va aggiornato in quanto prima valeva 4 mentre ora $C(1) = C(2) + 2 = 3$. Si elimina quindi 2 dalla coda e si pone $C^*(2) = 1$. Si ha ora che $Q = \{1, 3\}$ e si esamina il nodo 1 il cui unico successore è 3 per il quale non si deve aggiornare il costo in quanto ora risulta maggiore $C(3) = C(1) + 1 = 4 > 3$. Si elimina 1 dalla coda e si pone $C^*(1) = 3$. Infine $Q = \{3\}$ e quindi $C^*(3) = 3$. Si sono così ottenuti tutti i cammini minimi a partire da x_I.

2.2.1.2 Algortimo A-Star

L'algoritmo A^* è una variante della programmazione dinamica che tenta di ridurre il numero totale di stati esplorati incorporando una stima euristica del costo per raggiungere lo stato finale da un dato stato. Sia $C(x)$ il cost-to-come da x_I a x e $G(x)$ il cost-to-go da x a qualche stato in $\mathcal{X}_G$. Sebbene $C^*(x)$ si possa calcolare in modo incrementale tramite la programmazione dinamica, non ci sono possibilità di conoscere in anticipo il valore del cost-to-go ottimo G^*. È comunque spesso possibile ottenere una buona stima inferiore di G^*, l'obiettivo è quello di trovarne una che sia più vicina possibile a G^*. Sia $\hat{G}^*(x)$ tale stima.

La differenza tra l'algoritmo A^* e Dijkstra sta nella funzione utilizzata per ordinare la coda Q. Nell'algoritmo A^* si usa la somma $C(x') + \hat{G}^*(x')$, pertanto la coda di priorità è ordinata in base alle stime del costo ottimo tra x_I e $\mathcal{X}_G$. Se effettivamente $\hat{G}^*(x)$ è una stima inferiore del cost-to-go ottimo per ogni $x \in \mathcal{X}$ allora l'algoritmo A^* trova cammini minimi. Se $\hat{G}^*(x) = 0$ per ogni $x \in \mathcal{X}$ allora l'algoritmo A^* coincide con l'algoritmo di Dijkstra.

2.2.2 Altri metodi di ricerca generali

Analizziamo ora due altri particolari algoritmi di ricerca, un metodo di ricerca all'indietro e un approccio bidirezionale.

2.2.2.1 Algoritmo di ricerca all'indietro

Si consideri il caso in cui ci sia un solo stato di arrivo x_G, può essere più efficiente procedere all'indietro a partire da x_G per arrivare a x_I.

Sia $\mathcal{U}^{-1} = \{(x, u)|x \in \mathcal{X}, u \in \mathcal{U}\}$ l'insieme di tutte le coppie stato-ingresso, questo può anche essere visto come il dominio della funzione di transizione f. Sia invece $\mathcal{U}^{-1}(x') = \{(x, u) \in \mathcal{U}^{-1}|x' = f(x, u)\}$. Per semplicità denotiamo con u^{-1} una coppia stato-ingresso in qualche $\mathcal{U}^{-1}(x')$. Per ogni $u^{-1} \in \mathcal{U}^{-1}(x')$ esiste un unico stato $x \in \mathcal{X}$. Sia f^{-1} una *funzione di transizione degli stati all'indietro*, possiamo scrivere $x = f^{-1}(x', u^{-1})$. Lo schema per questo algoritmo è simile a quello proposto in 2.2.1.

Ricerca all'indietro

```
1  Q.Insert(x_G)
2  while Q not empty do
3        x' ←Q.GetFirst()
4        if  x = x_I
5            return SUCCESS
6        forall u^{-1} ∈ U^{-1}(x)
7            x ← f^{-1}(x, u^{-1})
8            if x not visited
9                Mark x as visited
10               Q.Insert(x)
11           else
12               Resolve duplicate x
13 return FAILURE
```

2.2.2.2 Algortimo bidirezionale

Negli algoritmi bidirezionali un albero viene generato a partire dallo stato iniziale x_I e uno a partire dallo stato finale x_G. La ricerca termina con successo quando i due alberi si incontrano, mentre fallisce se una delle code di priorità si esaurisce. Lo schema di questi algoritmi è la cominazione dei due schemi già riportati.

2.3 Pianificazione ottima discreta

Consideriamo ora il caso in cui non si è semplicemente interessati a trovare una traiettoria ammissibile ma si vuole determinare la traiettoria ammissibile che ottimizza un dato criterio come il tempo di percorrenza o l'energia consumata. Per poter estendere la formalizzazione introdotta nel paragrafo 2.1 devono essere introdotte altri concetti:

1 un indice associato al passo della pianificazione;

2 una funzione costo che rappresenta il costo accumulato durante l'esecuzione della traiettoria;

3 una azione di terminazione dell'algoritmo.

Si inizia con il considerare il caso in cui si cercano cammini ottimi di lunghezza fissata. Successivamente lo si estende al caso di cammini ottimi di lunghezza variabile. Infine verrà messo in relazione l'algoritmo proposto in questo paragrafo con l'algoritmo di Dijkstra proposto in 2.2.1.1.

Per semplicità considereremo come criterio di ottimizzazione quello della minimizzazione della funzione costo. Moltiplicando la funzione costo per il fattore -1 si ottiene un problema di massimizzazione.

2.3.1 Pianificazione discreta ottima a lunghezza fissata

Sia π_K una *pianificazione a K passi*, i.e. una sequenza $(u_1, u_2, \ldots, u_K)$ di K azioni di controllo. Dati π_K e x_I è possibile ricostruire, tramite la funzione di transizione f, la sequenza degli stati $x_I, x_2, \ldots, x_{K+1}$ dove $x_{k+1} = f(x_k, u_k)$.

Sia L una funzione costo a valori reali che sia additiva rispetto al passo K della pianificazione e che sia definita su una pianificazione a K passi π_K. Se si denota con F il *passo finale*, $F = K + 1$ si ha che la *funzione costo* è data da

$$L(\pi_K) = \sum_{k=1}^{K} l(x_k, u_k) + l_F(x_F).$$

Il termine $l_F(x_F)$ è tale che $l_F(x_F) = 0$ se $x_F \in \mathcal{X}_G$ e $l_F(x_F) = \infty$ altrimenti. In questo modo L è definita su ogni traiettoria, quelli che non consentono di raggiungere stati in $\mathcal{X}_G$ vengono penalizzati da un costo infinito.

Se si è interessati soltanto a traiettorie ammissibili di lunghezza K è sufficiente porre $l(x, u) \equiv 0$. Se invece si vuole minimizzare il numero di passi è sufficiente porre $l(x, u) \equiv 1$.

Gli algoritmi che descriveremo in seguito si basano sul principio di ottimalità secondo il quale ogni porzione di una traiettoria ottima è essa stessa ottima. Questo principio induce un algoritmo di tipo iterativo detto *value iteration* in grado di risolvere una vasta gamma di problemi e non soltanto quelli discussi in questo corso (ad esempio pianificazione con incertezze stocastiche o con misure degli stati non perfette). L'idea è quella di calcolare iterativamente le funzioni cost-to-go e cost-to-come ottime.

2.3.1.1 Backward value iteration

Sia G_k^* per $1 \leq k \leq F$ il costo che si accumula dal passo k al passo F lungo un cammino ottimo:

$$G_k^*(x_k) = \min_{u_k, \ldots, u_K} \left\{ \sum_{i=k}^{K} l(x_i, u_i) + l_F(x_F) \right\}. \tag{2.1}$$

Per $k = F = K + 1$ si ha $G_F^*(x_F) = l_F(x_F)$. Partendo da x_F si ottiene

$$G_K^*(x_K) = \min_{u_K} \{ l(x_K, u_K) + l_F(x_F) \} = \min_{u_K} \{ l(x_K, u_K) + G_F^*(f(x_K, u_K)) \},$$

che risulta facilmente calcolabile per ogni $x_K \in \mathcal{X}$.

Mostriamo ora come calcolare G_k^* a partire da G_{k+1}^*. La 2.1 può essere riscritta nel seguente modo

$$\begin{aligned} G_k^*(x_k) &= \min_{u_k} \min_{u_{k+1}, \ldots, u_K} \left\{ l(x_k, u_k) + \sum_{i=k+1}^{K} l(x_i, u_i) + l_F(x_F) \right\} && (2.2) \\ &= \min_{u_k} \left[l(x_k, u_k) + \min_{u_{k+1}, \ldots, u_K} \left\{ \sum_{i=k+1}^{K} l(x_i, u_i) + l_F(x_F) \right\} \right] && (2.3) \\ &= \min_{u_k} \left\{ l(x_k, u_k) + G_{k+1}^*(x_{k+1}) \right\}. && (2.4) \end{aligned}$$

L'ultimo membro della catena di uguaglianze dipende soltanto da x_k, u_k e G^*_{k+1}. Si prosegue sino ad ottenere G^*_1 da cui poi si ottiene $G^*(x_I)$ che rappresenta il costo ottimo per andare da x_G a x_I.

Per ottenere anche la sequenza di controlli ottima è necessario mantenere in memoria il controllo che minimizza la 2.2.

Esempio 1: Dato il grafo in figura 2.6, si consideri $x_I = a$, $\mathcal{X}_G = \{d\}$ e $K = 4$. Calcolando all'indietro

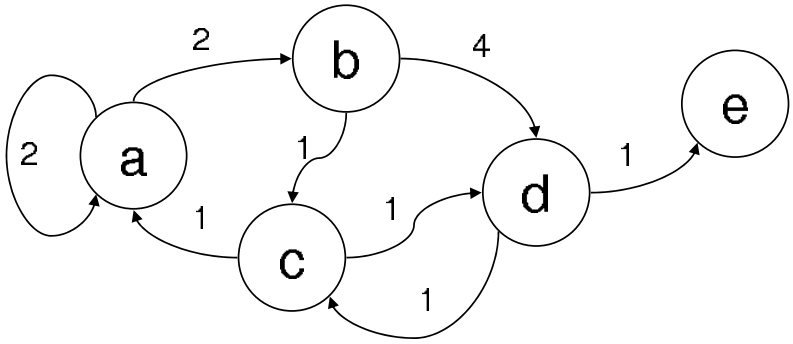


Figura 2.6: Determinare il cammino a costo minimo di 4 passi.

il cost-to-go ottimo si ottiene la seguente tabella.

	a	b	c	d	e
G^*_5	∞	∞	∞	0	∞
G^*_4	∞	4	1	∞	∞
G^*_3	6	2	∞	2	∞
G^*_2	4	6	3	∞	∞
G^*_1	6	4	5	4	∞

Infatti inizialmente $K + 1 = 5$ e $x_F = d$ per cui $G^*_5(d) = 0$ e tutti gli altri valgono ∞. Il nodo d in un passo si può raggiungere soltanto dai nodi b e c pertanto si aggiornano i rispettivi valori di G^*_4 a 4 e 1 rispettivamente. Questi nodi sono raggiungibili in un passo (e quindi d lo è in due passi) dai nodi a, b e d. Si procede in questo modo sino ad ottenere G^*_1. In figura 2.7 sono riportati i vari passi con i costi dei cammini. I costi ottimi per ogni passo sono riportati in grassetto. Il cost-to-go del nodo e non viene mai aggiornato e pertanto rimane infinito. Infatti dal nodo e non è possibile raggiungere mai il nodo d e tanto meno in 4 passi.

2.3.1.2 Forward value iteration

L'iterazione in avanti può essere usata per calcolare tutti i cammini ottimi per raggiungere ogni stato di $\mathcal{X}$ a partire da un x_I fissato. Sia C^*_k il *cost-to-come* ottimo dal passo 1 al passo k ottimizzato su

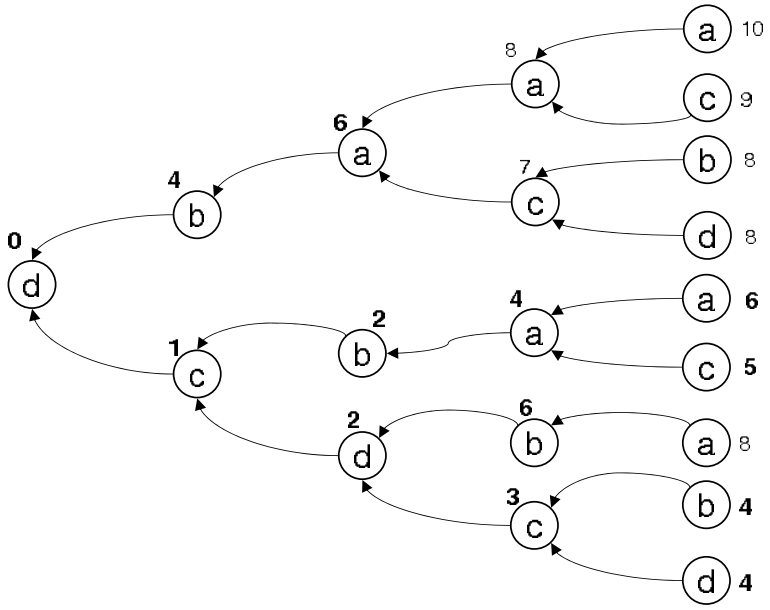


Figura 2.7: Procedimento per il calcolo del cammino a costo minimo di 4 passi all'indietro.

tutti i cammini di $k-1$ passi. Per escludere cammini che non partono da x_I si pone $C_1^*(x_I) = l_I(x_I)$ dove $l_I(x_I) = 0$ mentre $l_I(x) = \infty$ per $x \neq x_I$. In generale il cost-to-come ottimo è dato da

$$C_k^*(x_k) = \min_{u_1,\,\ldots,\,u_{k-1}} \left\{ l_I(x_1) + \sum_{i=1}^{k-1} l(x_i,\, u_i) \right\} \tag{2.5}$$

e al passo finale F si ha

$$C_F^*(x_F) = \min_{u_1,\,\ldots,\,u_K} \left\{ l_I(x_1) + \sum_{i=1}^{K} l(x_i,\, u_i) \right\} \tag{2.6}$$

Per ottenere una formula ricorsiva supponiamo di conoscere C_{k-1}^*:

$$C_k^*(x_k) = \min_{u^{-1} \in \mathcal{U}^{-1}(x_k)} \left\{ C_{k-1}^*(x_{k-1}) + l(x_{k-1},\, u_{k-1}) \right\}, \tag{2.7}$$

dove $x_{k-1} = f^{-1}(x_k,\, u_k^{-1})$ e $u_{k-1} \in \mathcal{U}(x_{k-1})$ è l'ingresso a cui corrisponde $u_k^{-1} \in \mathcal{U}^{-1}(x_k)$.

Esempio 2: Si consideri nuovamente il grafo in figura 2.6 ma si ponga $x_I = a$ e $K = 4$. Le funzioni cost-to-come ottime calcolate con l'iterazione in avanti sono riportate nella seguente tabella.

	a	b	c	d	e
C_1^*	0	∞	∞	∞	∞
C_2^*	2	2	∞	∞	∞
C_3^*	4	4	3	6	∞
C_4^*	6	6	5	4	7
C_5^*	6	6	5	6	5

In figura 2.8 sono riportati i vari passi con i costi dei cammini. I costi ottimi per ogni passo sono riportati in grassetto.

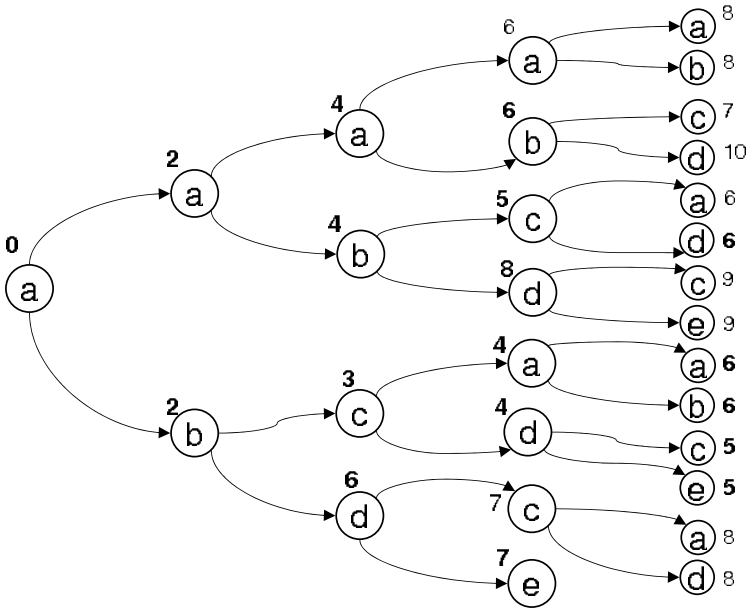


Figura 2.8: Procedimento per il calcolo del cammino a costo minimo di 4 passi in avanti.

2.3.2 Pianificazione discreta ottima a lunghezza variabile

I metodi visti nel paragrafo precedente si possono facilmente estendere al calcolo di traiettorie ottime di lunghezza non specificata. È necessario introdurre una *azione terminale* u_T tale che quando applicata ad uno stato x_k lo stato non cambierà per i passi successivi. In altre parole per ogni $i \geq k$, $u_i = u_T$, $x_i = x_k$ e $l(x_i, u_T) = 0$.

Con questa azione terminale è possibile utilizzare i metodi iterativi introdotti precedentemente con passo K e ottenere camini ottimi di lunghezza minore o uguale a K. Infatti se si ha un cammino ottimo in 2 passi si può applicare per tre passi successivi l'azione terminale e si ottiene un cammino ottimo in 5 passi per lo stesso stato.

Il passo successivo da compiere, nella estensione degli algoritmi, è quello di eliminare la dipendenza da K. Se si considera l'iterazione all'indietro, una volta calcolato G_1^* è possibile continuare il procedimento e calcolare G_0^*, G_{-1}^* eccetera. Il procedimento si interrompe nel momento in cui la situazione diventa *stazionaria* e cioè quando per ogni $i \leq k$ $G_{i-1}^*(x) = G_i^*(x)$ per tutti gli stati $x \in \mathcal{X}$. Nel momento in cui ci si trova in una sistuazione stazionaria il cost-to-go non dipende più dal particolare passo k. È importante osservare che se la funzione $l(x, u)$ non è mai negativa allora prima o poi ci si trova in condizione di stazionarietà.

Lo stesso risultato si ottiene nel caso di iterazione in avanti. Inoltre con entrambi gli approcci è possibile ottenere tramite l'algoritmo anche la sequenza dei controlli che determina la traiettoria ottima.

Esempio 3: Si consideri nuovamente il grafo in figura 2.6, le funzioni cost-to-come ottime calcolate con l'iterazione all'indietro sono riportate nella seguente tabella.

	a	b	c	d	e
G_0^*	∞	∞	∞	0	∞
G_{-1}^*	∞	4	1	0	∞
G_{-2}^*	6	2	1	0	∞
G_{-3}^*	4	2	1	0	∞
G_{-4}^*	4	2	1	0	∞

Si noti che dopo 4 passi si è in condizione di stazionarietà.

Nel caso dell'iterazione in avanti si consideri $x_I = b$:

	a	b	c	d	e
C_1^*	∞	0	∞	∞	∞
C_2^*	∞	0	1	4	∞
C_3^*	2	0	1	2	5
C_4^*	2	0	1	2	3
C^*	2	0	1	2	3

Anche in questo caso dopo 4 passi si è in condizione di stazionarietà.

2.3.3 Algoritmo di Dijkstra e programmazione dinamica

Il metodo iterativo in avanti e l'algoritmo di Dijkstra sono molto simili. In particolare però l'algoritmo di Dijkstra etichetta come morti i nodi per i quali il costo non verrà modificato e tali nodi non verranno più rivisitati dall'algoritmo. In generale però l'algoritmo di Dijkstra può risultare computazionalmente costoso nella gestione delle code.

Capitolo 3

Controllo Ottimo

3.1 Il Principio del Massimo

Consideriamo un sistema dinamico nonlineare nella forma piuttosto generale

$$\dot{x} = f(x,u), \quad x \in \mathbb{R}^n, \ u \in \mathbb{R}^m, \ x(0) = x_0$$

Consideriamo anche un indice obiettivo

$$J = \Psi(x(T)) + \int_0^T L(x(t),u(t))dt$$

che considera i valori ottenuti dallo stato ad un tempo finale T, oltreché l'andamento dello stato e del controllo lungo tutto l'intervallo tra $t = 0$ e $t = T$. Si noti che la funzione obiettivo è interamente determinata, per un dato sistema dinamico e per date condizioni iniziali, dalla funzione di ingresso $u(t)$. Consideriamo il problema di massimizzare J, che rappresenta un *funzionale* in quanto funzione di funzioni, rispetto alle possibili scelte del controllo. In generale, i valori istantanei del controllo potranno essere soggetti a restrizioni (ad esempio, valori limitati in modulo), nel qual caso restringeremo opportunamente $u \in \mathcal{U}$.

Quello posto è un problema di calcolo variazionale, cioè un problema di ottimizzazione in cui la incognita è una funzione invece che semplicemente una variabile incognita. Un particolare ingresso $\hat{u}(t)$ è ottimo se $J(\hat{u}) \geq J(u)$, $\forall u \in \mathcal{U}$. Essendo la minimizzazione di J sottoposta al vincolo tra l'andamento di $x(t)$ e quello di $u(t)$ espresso dalla dinamica, possiamo procedere, secondo una tecnica analoga a quella dei moltiplicatori di Lagrange nel caso di ottimizzazione di funzioni con vincoli, a scrivere un indice modificato

$$J_0 = J - \int_0^T p^T(\dot{x} - f(x,u))dt, \quad p \in \mathbb{R}^n$$

per il quale vale ovviamente $J_0 = J$, $\forall t$, $\forall u$, qualsiasi sia la scelta del moltiplicatore $p \in \mathbb{R}^n$, che potrà anche essere variabile nel tempo (cioè una funzione $p : [0,T] \to \mathbb{R}^n, t \mapsto p(t)$).

Riscriviamo

$$\begin{aligned} J_0 = \quad & \Psi(x(T)) + \int_0^T \left[L(x,u) + p^T f(x,u) - p^T \dot{x})\right] dt \\ & := \Psi(x(T)) + \int_0^T \left[H(p,x,u) - p^T \dot{x}\right] dt \end{aligned}$$

dove si è definito implicitamente il funzionale

$$H(p,x,u,t) = p^T f(x,u) + L(x,u,t)$$

che viene detto *Hamiltoniano* del problema.

Cercando di dare condizioni necessarie affinché un ingresso $\hat{u}(t)$ sia ottimo, consideriamo il problema localmente, cioè confrontiamo l'indice ottenuto da $\hat{u}(t)$ rispetto a quello ottenuto da funzioni $u(t)$ che differiscano "poco" da $\hat{u}(t)$. Considereremo quindi funzioni u per cui valga $\|\hat{u} - u\| := \max_i \int_0^T |\hat{u}_i(t) - u_i(t)|dt < \epsilon$, con ϵ opportunamente piccolo (si noti che sono ammesse differenze anche grandi tra le componenenti di ingresso, purché per tempi brevi).

Per la continuità delle soluzioni delle equazioni differenziali ordinarie, anche le soluzioni $x(t, x_0, u)$ differiranno poco dalla soluzione $x(t, x_0, \hat{u})$, e scriveremo $x(t, x_0, \hat{u}) - x(t, x_0, u) = \delta x(t)$, con $\|\delta x(t)\|$ infinitesimo dello stesso ordine di ϵ. La funzione obiettivo è corrispondentemente modificata da

$$\begin{aligned}\delta J_0 = \quad & \Psi(x(T) + \delta x(T)) - \Psi(x(T)) \\ & + \int_0^T [H(p, x + \delta x, u) - H(p, x, \hat{u})]\, dt \\ & - \int_0^T \left[p^T(\dot{x} + \dot{\delta x}) - p^T\dot{x}\right] dt\end{aligned}$$

Approssimando al primo ordine, e indicando con un pedice le derivate parziali, si ha

$$\Psi(x(T) + \delta x(T)) - \Psi(x(T)) \approx \Psi_x(x(T))\delta x(T),$$

e

$$\begin{aligned}&\int_0^T [H(p, x + \delta x, u) - H(p, x, \hat{u})]\, dt \\ &= \int_0^T [H(p, x + \delta x, u) - H(p, x, u) + H(p, x, u) - H(p, x, \hat{u})]\, dt \\ &\approx \int_0^T [H_x(p, x, u)\delta x + H(p, x, u) - H(p, x, \hat{u})]\, dt.\end{aligned}$$

Usando invece la regola di integrazione per parti, si ha che il terzo addendo in δJ_0 vale

$$\begin{aligned}\int_0^T p^T\dot{\delta x}dt = \quad & \left[p^T\delta x\right]_0^T - \int_0^T \dot{p}^T\delta x dt \\ & = p(T)^T\delta x(T) - p(0)^T\delta x(0) - \int_0^T \dot{p}^T\delta x dt.\end{aligned}$$

Osservando che $\delta x(0) = 0$ (le variazioni del controllo non hanno influenza sulle condizioni iniziali), possiamo scrivere

$$\begin{aligned}\delta J_0 \approx \quad & \\ & \left[\Psi_x(x(T)) - p^T(T)\right]\delta x(T) \\ & + \int_0^T \left[H_x(p, x, u) + \dot{p}^T\right]\delta x dt \\ & + \int_0^T [H(p, x, u) - H(p, x, \hat{u})]\, dt\end{aligned}$$

a meno di infinitesimi di ordine superiore rispetto a ϵ. Possiamo adesso semplificare δJ_0 usando la libertà che ci è concessa nella scelta di $p(t)$. Ponendo infatti

$$\dot{p}(t) = -H_x^T(p, x, u) \quad \text{e} \quad p(T) = \Psi_x^T(x(T)),$$

si ottiene

$$\delta J_0 = \int_0^T [H(p, x, u) - H(p, x, \hat{u})]\, dt.$$

Si noti che le scelte fatte per $p(t)$ equivalgono a definire una equazione differenziale ordinaria *aggiunta* al problema, con condizioni non iniziali come consueto, bensì terminali.

Se $\hat{u}(t)$ è ottima, come supposto, allora deve essere $\delta J_0 < 0$, $\forall u(t)$ nell'insieme considerato. Questo implica che per ogni t, valga

$$H(p, x, u) \leq H(p, x, \hat{u}).$$

Questa affermazione, molto più forte della precedente, discende dal fatto che, se esistesse una u per la quale, anche in un solo istante t^*, valesse $H(p,x,u(t^*)) > H(p,x,\hat{u}(t^*))$, allora si potrebbe costruire un nuovo ingresso $w(t) = \hat{u}(t)$, $\forall t \neq t^*$, ma $w(t^*) = u(t^*)$, per la quale risulterebbe $\delta J_0 > 0$, contro l'ipotesi che $\hat{u}$ sia ottima.

E' ovvio che la relazione $H(p,x,u) \leq H(p,x,\hat{u})$, quando fossero noti lo stato ed il co-stato ad un tempo t, permetterebbe di trovare il valore ottimo u con la soluzione di un normale problema di massimizzazione di una funzione rispetto ad una variabile. In particolare, se gli ingressi non sono soggetti a vincoli, una condizione necessaria affinché $\hat{u}(t)$ sia ottimo è che esso sia un estremo di $H(x,p,u)$, cioè che

$$H_u(x,p,u)|_{\hat{u}} = 0.$$

Un contributo importante che generalizza la applicabilità di questa osservazione al caso (praticamente molto importante) in cui i valori del controllo siano limitati in un insieme compatto $u(t) \in \mathcal{U}$, è il seguente

> *Principio del Massimo* di Pontryagin: se $\hat{u}(t)$ è il controllo ottimo, allora $H(x(t),p(t),\hat{u}(t))$ assume il valore massimo tra quelli ottenuti da $u(t) \in \mathcal{U}$

Si osservi esplicitamente come i massimi di una funzione continua su un compatto possono essere ottenuti non solo nei punti estremali ma anche sulla frontiera dell'insieme.

Riassumendo, abbiamo trovato che:

Se $\hat{u}(t)$ e $x(t)$ sono la funzione di ingresso e la corrispondente traiettoria di stato soluzioni del problema di controllo ottimo sopra definito, allora esiste una traiettoria (detta di co-stato) $p(t)$ che soddisfa le seguenti condizioni:

$\dot{x} = f(x,u);$	dinamica dello stato
$x(0) = x_0;$	condizioni iniziali in x
$\dot{p} = -f_x^T(x,u)p(t) - L_x^T(x,u);$	dinamica del co-stato
$p(T) = \Psi_x^T(x(T));$	condizioni finali sul co-stato

ed inoltre vale

$$H(x,p,\hat{u}) \geq H(x,p,u),\ \forall u \in \mathcal{U}.$$

Questo sistema di equazioni definisce completamente l'ottimo, nel senso che si hanno tante equazioni quante incognite: queste ultime sono le $2n+m$ funzioni $u(t)$, $x(t)$ e $p(t)$, determinate dalle $2n$ equazioni differenziali ordinarie dello stato e del co-stato, con le loro rispettive condizioni agli estremi, e dalle m condizioni di massimizazzione dell'Hamiltoniano.

La soluzione di questo sistema di equazioni non è peraltro facile nel caso generale. Una delle cause principali di tali difficoltà deriva dal fatto che le condizioni agli estremi sono miste iniziali e finali. Anche le soluzioni numeriche possono risultare molto impegnative.

Si dà talvolta una formulazione diversa del problema del controllo ottimo, detta Hamiltoniana, che gode di una maggiore compattezza ed eleganza. Si introduce un nuovo stato definito da

$$\dot{x}_L(t) = L(x(t),u(t)), \quad x_L(0) = 0$$

e lo si giustappone a x in un nuovo vettore $n+1$ dimensionale $x_e^T = [x^T, x_L^T]$, talché l'indice obiettivo diviene

$$J = \Psi(x(T)) + x_L(T) := \Phi(x_e(t))$$

La funzione hamiltoniana viene parimenti ridefinita da

$$H = p_e^T(t)\dot{x}_e(t) = p_L^T(t)\dot{x}_L(t) + p^T(t)\dot{x}(t)$$

da cui si ha

$$\begin{cases} \dot{x}_e(t) = \left(\frac{\partial H}{\partial p_e}\right)^T & x_e(0) = \begin{bmatrix} 0 \\ x(0) \end{bmatrix} \\ \dot{p}_e(t) = -\left(\frac{\partial H}{\partial x_e}\right)^T & p_e(T) = \begin{bmatrix} 1 \\ \Psi_x(x(T)) \end{bmatrix} \end{cases}$$

che definiscono interamente il problema assieme alla condizione di massimizzazione dell'hamiltoniano $\hat{u} = \arg\ \max_{u\in\mathcal{U}} H(x,p,u)$.

Si noti che, non essendo H funzione esplicita di x_L, dalle equazioni differenziale per il costato esteso si ricava $\dot{p}_L = 0$, $p_L(T) = 1$, da cui $p_L(t) \equiv 1$, per cui la funzione hamiltoniana coincide con quella precedentemente definita.

Sussiste infine la notevole relazione

$$\begin{aligned} \tfrac{d}{dt}H(x,p,t) &= H_p^T\dot{p} + H_x^T\dot{x} + H_t \\ &= -H_p^T H_x + H_x^T H_p + H_t = H_t \end{aligned}$$

da cui, se l'Hamiltoniano non dipende esplicitamente dal tempo (ovvero se il costo e la dinamica sono tempo-invarianti), si ha che, in corrisopndenza di traiettorie ottimali, l'Hamiltoniano stesso è costante, vale a dire è *un integrale primo del moto*.

Vi sono ovviamente altre possibili posizioni dei problemi di controllo ottimo.

In alcuni casi non ci si accontenta di pesare la distanza della configurazione finale da un valore desiderato come fatto con $\Psi(x(T))$, ma si vuole imporre esattamente a $x(T)$ un dato valore. In tal caso, la caratterizzazione sopra fornita della soluzione è ancora valida, laddove si rimuovano le condizioni finali sul co-stato (il numero di equazioni totale non cambia). In problemi in cui solo alcune componenti dello stato siano assegnate al tempo finale, saranno assegnate condizioni terminali solo alle componenti del co-stato di indice diverso da quelle.

Un altro caso di particolare interesse è quello in cui il tempo finale T sia libero, e rappresenta quindi una ulteriore variabile da determinare. Si noti dalla espressione dell'indice obiettivo

$$J_0 = \Psi(x(T)) + \int_0^T \left[H(p,x,u) - p^T\dot{x}\right] dt$$

che, in corrispondenza di un valore ottimo $\hat{T}$, la variazione di J_0 dovuta ad una modifica dell'estremo superiore di integrazione deve essere nulla.

Valendo la *condizione di trasversalità* al tempo finale $p^T(T)\dot{x}(T) = 0$, si trova la ulteriore condizione necessaria per il tempo finale ottimo

$$H\left(x(\hat{T}), p(\hat{T}), u(\hat{T})\right) = 0.$$

3.2 Esempi

3.2.1 Sistemi Meccanici Hamiltoniani

Come primo esempio, per mostrare la generalità delle relazioni trovate, useremo la tecnica variazionale appena descritta per ricavare le equazioni dinamiche del moto di un sistema conservativo, descritto da coordinate generalizzate q.

Si consideri dunque un sistema con Lagrangiana $L(q,\dot{q}) = T(q,\dot{q}) - U(q)$, dove $T(q,\dot{q})$ è l'energia cinetica, e $U(q)$ l'energia potenziale del sistema. Ci si propone di trovare la legge del moto di questo sistema che minimizza l'integrale della Lagrangiana, seondo quello che in fisica viene detto "principio di minima azione". Assimiliamo quindi le velocità da determinare alla funzione di controllo incognita u in un problema di controllo ottimo, cioè poniamo $\dot{q} = u$, e scriviamo

$$J = \int_0^T (T(q,u) - U(q))dt$$

ovvero

$$H(q,u,p) = T(q,u) - U(q) + p^T u$$

da cui

$$\dot{p}^T = -\frac{\partial H}{\partial q} = -\frac{\partial L}{\partial q},$$
$$0 = \frac{\partial H}{\partial u} = \frac{\partial L}{\partial u} + p^T.$$

Differenziando rispetto al tempo e combinando le due equazioni si ottiene

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = 0,$$

ovvero le equazioni di Eulero-Lagrange per il moto di sistemi conservativi.

Se il lagrangiano L non è funzione esplicita del tempo, H è un integrale primo, quindi è una costante del moto. Osservando che

$$H = T - U - \frac{\partial T}{\partial \dot{q}}\dot{q}$$

e che l'energia cinetica è una forma quadratica omogenea delle velocità, del tipo $T = \dot{q}^T I(q)\dot{q}$, e che quindi $\frac{\partial T}{\partial \dot{q}}\dot{q} = 2T$, si ottiene

$$-H = T + U = cost.$$

cioè, l'energia meccanica si conserva nel moto di un sistema conservativo.

Con piccole modifiche del procedimento precedente, è possibile trattare il caso in cui siano presenti forze generalizzate non conservative Q_{nc}, arrivando alla nota equazione

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = Q_{nc}.$$

3.2.2 Percorso minimo tra due punti

Ancora a livello illustrativo, si consideri il problema di trovare la più breve curva nel piano che unisca due punti dati, ad esempio la origine del piano con un punto di coordinate $(1,1)$. Questo problema può essere scritto in termini di controllo ottimo con alcuni semplici artifici.

Poniamo le coordinate del piano uguali a (t,x), e imponiamo le condizioni iniziali $x(t=0) = 0$, $x(t=1) = 1$. Sia inoltre $\dot{x}(t) = u(t)$ la pendenza della curva da determinare. La lunghezza dell'arco infinitesimo di curva corrispondente ad un incremento dt vale $\sqrt{dx^2 + dt^2}$ ovvero $\sqrt{(1+u^2)}dt$. Scriviamo quindi

$$\begin{aligned} \dot{x} &= u \\ x(0) &= 0 \\ x(1) &= 1 \\ J &= \int_0^1 \sqrt{1+u^2}dt \end{aligned}$$

Si noti che il problema è qui di minimizzazione, e non di massimizzazione come discusso in precedenza. Ciò non altera sostanzialmente la natura del problema: si può procedere o cambiando il segno del funzionale J e massimizzando, ovvero semplicemente procedendo come detto in precedenza eccetto per la ricerca del controllo ottimo, che sarà quello che minimizza l'Hamiltoniano.

L'Hamiltoniano del problema di minimizzazione vale

$$H = p^T u + \sqrt{1 + u^2}$$

L'equazione aggiunta è ovviamente $\dot{p} = H_x = 0$. Essendo fissato il valore terminale di $x(1)$, non è fissata la $p(T)$ (che dovrebbe essere determinata dalle altre condizioni, se necessario). Sappiamo comunque che si avrà $p(t) = cost$.

Anche senza conoscere p, né risolvere $H_u = 0$, osserviamo che l'unico termine in H che dipende dal tempo è u stesso. Dovendo $u(t)$ minimizzare H per ogni t, ne risulta che $\hat{u}(t) = u = cost$. Quindi la pendenza della curva è costante, cioè la curva più breve è un segmento di retta. Trovatane la natura , la specifica soluzione si trova a partire dalle condizioni ai tempi iniziale e finale: si tratta ovviamente della retta passante per i punti dati.

3.2.3 Posizionamento di una massa inerziale in tempo minimo

Consideriamo il problema di portare un corpo di massa $m = 1$ mobile su una retta senza attrito, da una posizione iniziale $x(0) = x_0$ all'origine nel minimo tempo possibile. Naturalmente, il problema ha senso solo se la forza con cui si può agire sull massa è limitata.

Scrivendo la dinamica $\ddot{x} = u$ in forma di stato, si vuole dunque minimizzare il problema

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \\ x(0) &= x_0 \\ x(T) &= 0 \\ -1 \leq\ & u \leq 1 \\ J &= \int_0^T 1 dt \end{aligned}$$

L'Hamiltoniano vale $H = 1 + p_1 x_2 + p_2 u$, da cui immediatamente si ha

$$\begin{aligned} \dot{p}_1 = -\tfrac{\partial H}{\partial x_1} = 0 \quad &\Rightarrow \quad p_1 = const. \\ \dot{p}_2 = -\tfrac{\partial H}{\partial x_2} = -p_1 \quad &\Rightarrow \quad p_2(t) = p_2^f + p_1(T - t) \end{aligned}$$

Il controllo ottimo è dunque quello che minimizza

$$H = 1 + p_1 x_2 + p_2(t) u$$

e quindi vale

$$\begin{cases} u = 1, & p_2 < 0; \\ u = -1, & p_2 > 0; \end{cases}$$

La condizione dei problemi a tempo minimo $H(t_f) = 0$, impone poi che $p_2^f u(T) = -1$ (si ricordi che $x(T) = 0$), quindi

$$\begin{cases} p_2^f > 0, \ u(T) < 0 \\ \text{ ovvero} \\ p_2^f < 0, \ u(T) > 0 \end{cases}$$

Si noti che il controllo ottimo non è definito negli istanti in cui si ha $p_2(t) = 0$. D'altronde, l'andamento lineare di $p_2(t)$ mostra che, eccettuato il caso in cui fosse $p_1 = 0$ e $p_2 = p_2^f = 0$, che è da escludere, si ha $p_2(t) = 0$ solo per un valore isolato $t = t^*$ nell'intervallo $[0, T]$: il controllo ottimo è quindi discontinuo in t^*.

Si osserva anche che il segno del controllo può cambiare una sola volta nel corso di una esecuzione del controllo.

Questo tipo di controllo, che usa solo i valori massimo e minimo dell'intervallo ammissibile, viene detto *bang-bang*. La funzione $p_2(t)$, i cui attraversamenti dello zero stabiliscono le commutazioni del valore del controllo, viene detta funzione di *switching*.

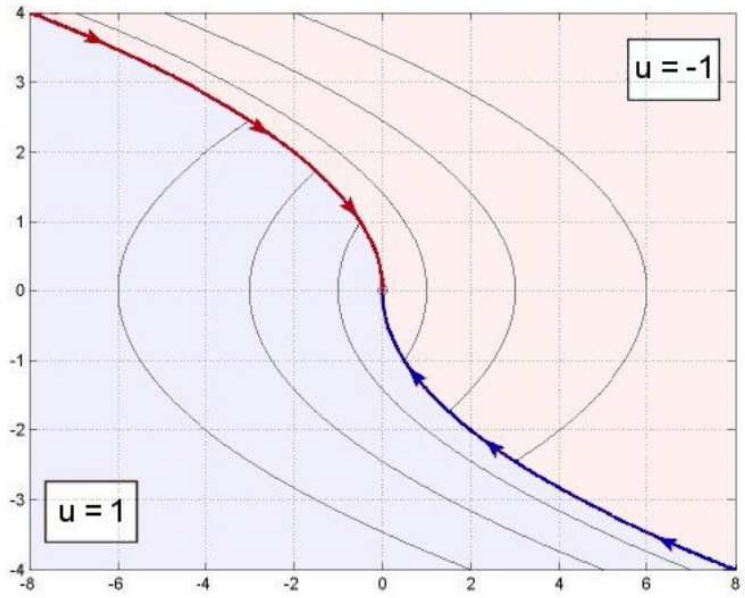
Il controllo ottimo risulta quindi in una fase di accelerazione massima seguita da una fase di decelerazione massima, o viceversa, a seconda delle condizioni iniziali.

La curva di switching può essere disegnata esplicitamente nel piano di stato, integrando all'indietro le equazioni del moto ottimo nei due casi:

a) $u(t_f) = -1 \Rightarrow x_2(t) = t_f - t$, $x_1(t) = -\frac{(t_f - t)^2}{2} \Rightarrow x_1 = -x_2^2/2$

b) $u(t_f) = 1 \Rightarrow x_2(t) = t - t_f$, $x_1(t) = -\frac{(t - t_f)^2}{2} \Rightarrow x_1 = x_2^2/2$

La curva di switching è data dunque da due rami di parabola, uniti nell'origine.



Le curve ottime sono anch'esse archi di parabola, paralleli alle precedenti.

3.2.4 Massimo spostamento di una massa con costo quadratico del controllo

Consideriamo ancora una massa unitaria in moto rettilineo, ma sottoposta adesso a una spinta il cui costo è proporzionale al quadrato della intensità. Si desidera massimizzare la distanza raggiunta dalla massa, contemperando però tale obiettivo con la riduzione del costo, secondo opportuni pesi relativi.

Scriviamo dunque

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \\ x(0) &= 0 \\ J &= x_1(T) - \int_0^t w\frac{u^2}{2}dt, \end{aligned}$$

dove w rappresenta il peso del costo del controllo relativamente al valore dell'obiettivo da massimizzare. Si ha facilmente

$$H = p_1 x_2 + p_2 u - w\frac{u^2}{2}$$

e le equazioni aggiunte

$$\begin{aligned} \dot{p}_1 &= -H_{x_1} = 0, \qquad p_1(T) = \Psi_{x_1}(x(T)) = 1; \\ \dot{p}_2 &= -H_{x_2} = -p_1, \quad p_2(T) = \Psi_{x_2}(x(T)) = 0. \end{aligned}$$

da cui immediatamente $p_1(t) \equiv 1$ e $p_2(t) = T - t$. Dalla massimizzazione dell'hamiltoniano risulta quindi

$$H_u = p_2 - wu \;\Rightarrow \hat{u} = \frac{1}{w}(T - t).$$

Il controllo ottimo decresce quindi linearmente nel tempo.

3.2.5 Percorsi minimi di veicoli a raggio di sterzo limitato

Consideriamo il modello cinematico di un veicolo su ruote

$$\begin{cases} \dot{x}(t) = u\cos\theta(t) \\ \dot{y}(t) = u\sin\theta(t) \\ \dot{\theta}(t) = \omega(t), \end{cases}$$

dove $\xi(t) = (x(t),\, y(t),\, \theta(t))$ rappresentano le coordinate del veicolo e la direzione di moto corrente, u e ω rappresentano, rispettivamente, le velocità lineari e angolari. Si considererà il caso in cui il veicolo proceda a velocità costante $u = U > 0$.

La limitatezza dell'angolo di sterzata del veicolo si può modellare con una limitazione della velocità angolare $|\omega| \leq \frac{U}{R}$. Si suppongano note una configurazione iniziale del veicolo $\xi_i = (x_i,\, y_i,\, \theta_i)$ e una configurazione finale $\xi_f = (x_f,\, y_f,\, \theta_f)$.

Vogliamo determinare il cammino di lunghezza minima tra le due configurazioni.

Il problema di controllo ottimo è posto come segue:

$$\begin{cases} \min J = \int_0^T 1dt, \\ \dot{x}(t) = U\cos\theta(t), \\ \dot{y}(t) = U\sin\theta(t), \\ \dot{\theta}(t) = \omega(t), \\ |\omega| \leq \frac{U}{R}, \\ \xi(0) = \xi_i, \\ \xi(T) = \xi_f. \end{cases} \tag{3.1}$$

L'Hamiltoniano del sistema risulta

$$H(p,\, \xi) = 1 + p_1 U\cos(\theta) + p_2 U\sin(\theta) + p_3\omega$$

Per il principio del minimo di Pontryagin, si ha che i controlli ottimi sono quelli che minimizzano l'Hamiltoniano

$$\hat{\omega} = \arg \min_{\omega \in [-\frac{U}{R}, \frac{U}{R}]} H(p, \xi).$$

Una condizione necessaria per l'ottimalità è:

$$\dot{p} = -\frac{\partial H}{\partial \xi}$$

Esplicitando la condizione sulle derivate del co-stato si ha che

$$\begin{aligned}
\dot{p}_1 &= -\tfrac{\partial H}{\partial x} = 0 \Rightarrow p_1 = const. := d\cos\phi \\
\dot{p}_2 &= -\tfrac{\partial H}{\partial y} = 0 \Rightarrow p_2 = const. := d\sin\phi \\
\dot{p}_3 &= -\tfrac{\partial H}{\partial \theta} = p_1 U \sin\theta - p_2 U \cos\theta = U d \sin(\theta - \phi)
\end{aligned}$$

Dalla condizione di ottimalità si ha che, all'interno dell'intervallo $-\frac{U}{R} < \omega < \frac{U}{R}$, deve valere:

$$\frac{\partial H}{\partial \omega} = p_3 = 0$$

Quindi se, lungo una traiettoria ottima, si ha controllo non saturato $|\omega| < \frac{U}{R}$, allora $\dot{p}_3 = U d \sin(\theta - \phi) = 0$ da cui segue che $\theta = \phi \pm \pi$, quindi θ costante. Si ottiene quindi che, se $|\omega| \neq \frac{U}{R}$, allora $\omega = 0$, e il tratto percorso dal veicolo è rettilineo.

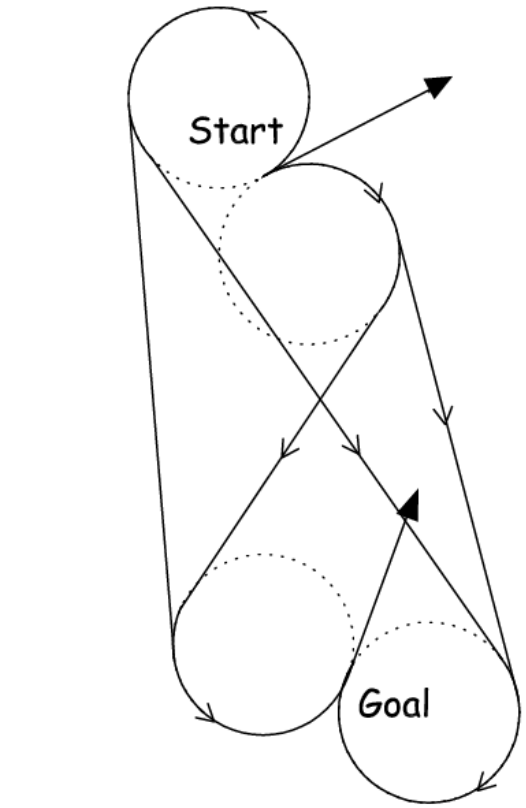
Altrimenti, essendo l'Hamiltoniano funzione lineare di ω, il suo minimo non potrà trovarsi che sui bordi dell'intervallo ammissibile per ω, e si avrà

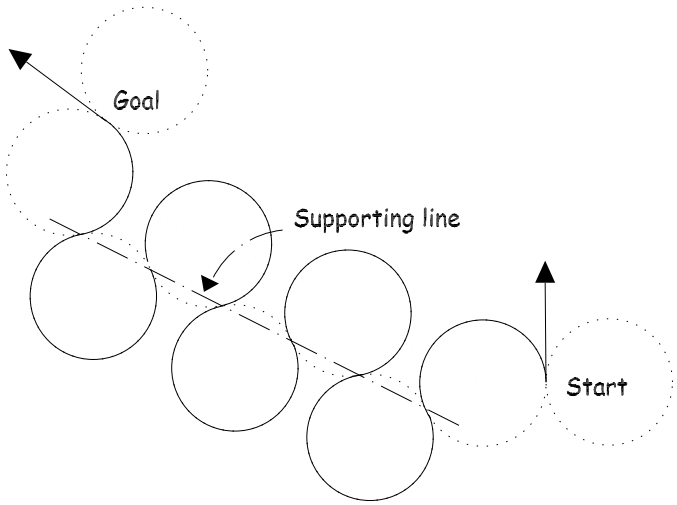
$$\begin{cases}
U d \sin(\theta - \phi) > 0 \Rightarrow \omega = -\frac{U}{R} \\
U d \sin(\theta - \phi) < 0 \Rightarrow \omega = \frac{U}{R}
\end{cases}$$

In queste circostanze, quindi, il veicolo percorre tratti curvilinei di circonferenze di raggio massimo, curvando a destra o a sinistra a seconda del segno della funzione di switching $U d \sin(\theta - \phi)$.

• Denotiamo con C_R e C_L un tratto di traiettoria corrispondente ad un arco di circonferenza di raggio minimo (R) e pewrcorso in senso orario o antiorario, e con S un tratto rettilineo.

• Le traiettorie ottime vanno quindi cercate tra "parole" candidate costruite con le "lettere" C_R, C_L, S. Ogni tratto ha una propria lunghezza, che deve essere trovata sulla base delle condizioni al contorno e di ottimalità.





• Nel 1957, Dubins ha dimostrato il seguente teorema: Ogni curva continua con derivata continua e derivata seconda continua a tratti, che colleghi due punti orientati nel piano con curvatura limitata, è di lunghezza non inferiore a quella di una curva di tipo $C_RC_LC_R$, o $C_LC_RC_L$, oppure CSC, con $C \in \{C_R, C_L\}$.

• Reeds e Shepp hanno esteso il risultato al caso in cui U possa variare in un intervallo $|U| \leq 1$, nel qual caso i percorsi minimi possono essere scritti come 46 diverse possibili parole di non più di cinque lettere tra C_R, C_L, S, intervallate da non più di due inversioni di velocità.