UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria dell'Informazione

Tesina di laurea triennale

INTRODUZIONE ALLA TECNICA DI PROGRAMMAZIONE DINAMICA PER IL CONTROLLO OTTIMO

Relatore: Prof. SANDRO ZAMPIERI
Laureando: GIULIANO ZAMBONIN

ANNO ACCADEMICO 2011-2012

Indice

Introduzione

La tecnica risolutiva della programmazione dinamica viene tipicamente applicata a problemi di ottimizzazione per i quali è necessario attuare delle scelte in sequenza a diversi livelli ed istanti di tempo per raggiungere una soluzione ottimale in riferimento ad una funzione di costo (o funzione obiettivo).

In un generico problema di ottimizzazione si ha l'obbiettivo di trovare il valore di *u* (variabile di decisione) per cui è possibile ottenere $\min_{u \in U} g(u)$ con *g(u)* funzione di costo e U l'insieme di vincolo.

Nei problemi stocastici il costo viene fornito come aspettazione di una variabile aleatoria *w* per cui $g(u) = E_{w}[G(u,w)]$. La programmazione dinamica può quindi affrontare problemi stocastici complessi dove l'informazione relativa a *w* diviene disponibile in passi successivi.

La teoria del controllo ottimo affonda le proprie radici nel calcolo delle variazioni (minimizzazione di funzionali[1]) ed è alla base di tecnologie importanti nel campo dell'ingegneria.

Obiettivo del controllo ottimo è determinare i segnali di controllo tali per cui il sistema da controllare soddisfi determinati vincoli fisici e allo stesso tempo renda minimo (o massimo) un indice predefinito per misurarne le prestazioni.

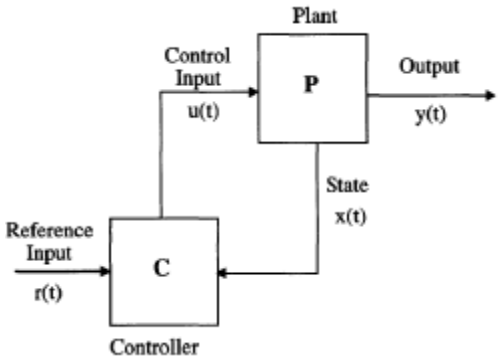

***Figura 1**: Schema generale di controllo. Lo scopo è trovare u(t) che porti l'impianto dallo stato iniziale allo stato finale con vincoli su u(t) e sullo stato x(t) e che allo stesso tempo estremizzi un indice (J).*

La formulazione di un problema di controllo ottimo richiede:

1. un modello che descrive il comportamento del sistema dinamico da controllare;
2. indice di comportamento J (criterio di ottimalità, funzione di costo o funzione obiettivo) che tiene conto delle specifiche desiderate e delle esigenze di progetto;
3. vincoli fisici e condizioni su stati e controllo (opzionali).

1 Vedere nota a pag 24.

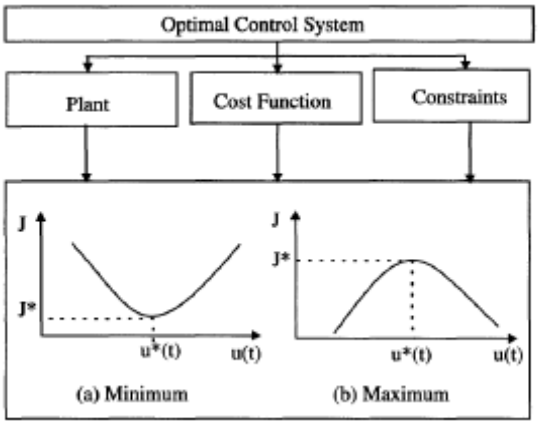

Figura 2*: schema per la formulazione di un problema di controllo ottimo.*

Scopo di questa tesi è introdurre la programmazione dinamica come metodo per il calcolo del controllo ottimale in forma chiusa[2].

Nel seguito vengono esposti i principi alla base dell'ottimizzazione dinamica ed alcuni esempi applicativi rilevanti come il problema del percorso minimo *(shortest path problem)* per il quale esiste una vasta gamma di tecniche risolutive oltre a quelle che vengono discusse in questo elaborato.

Il legame tra l'algoritmo di programmazione dinamica (DP) e i teoremi del controllo ottimo viene approfondito con l'analisi dettagliata di problemi riguardanti sistemi deterministici a tempo continuo.

L'intento è quello di soffermarsi sui principi alla base dell'algoritmo DP come tecnica efficace per i problemi di controllo ottimo fornendo una panoramica di ambiti di applicazione differenti in cui tale metodo si rivela valido.

I concetti esposti in questo elaborato sono tratti dai primi capitoli del testo *Dynamic programming and Optimal Control (vol. 1)*di Dimitri P. Bertsekas rispettandone la struttura: la parte iniziale riassume gli aspetti fondanti del procedimento DP, segue una sezione riguardante l' utilizzo nel problema del percorso minimo e la parte finale in relazione ai principi del controllo ottimo con esempi di approfondimento.

2 Tramite unica espressione.

CAPITOLO 1

ALGORITMO DI PROGRAMMAZIONE DINAMICA

1.1 Modello di controllo ottimo per un sistema dinamico

Molte tipologie di problemi possono essere trattati con la tecnica di programmazione dinamica e a prescindere dallo specifico ambito applicativo è possibile formulare un modello generale valido per sistemi dinamici a tempo discreto per i quali viene stabilita una funzione costo additiva[3].

Sistema a tempo discreto: $x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0,1,\ldots,N-1$ con

-k : tempo discreto

-x_k: stato, sintetizza informazione rilevante per la futura ottimizzazione

-u_k: controllo o decisione selezionata al tempo k[4]

-w_k: disturbo o rumore

-N : numero di applicazioni

per la presenza del parametro w_k il costo ($g_k(x_k, u_k, w_k)$) assume la forma di una variabile aleatoria (costo atteso). Con $g_N(x_N)$ costo terminale di fine processo si ottiene

$$E[g_N(x_N) + \sum_{k=0}^{N-1} g_K(x_k, u_k, w_k)]$$

Rilevante è la distinzione tra minimizzazione di tipo *closed-loop* e *open-loop.*

Nella minimizzazione *open-loop* vengono selezionati tutti i controlli $u_0, \ldots., u_{N-1}$ al tempo 0, nel tipo *closed-loop* la scelta per u_k viene posticipata al momento k una volta noto x_k ;

l'ottimizzazione di questo tipo prevede la scelta di $u_k \in U_k(x_k)$ per ogni k ed ogni possibile valore di x_k .

Il problema di base consiste nella ricerca di una legge di controllo ottima (a minimo costo) del tipo $\pi = \{\mu_0, \ldots.., \mu_{N-1}\}$ che definisce la sequenza di funzioni che mappano gli stati nei controlli ($u_k = \mu_k(x_k)$)[5].

Dato uno stato iniziale x_k e un insieme π , lo stato evolve con equazione di sistema

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \qquad k = 0,1,\ldots,N-1 \ ,$$

con $g(k), k = 0,1,\ldots, N-1$ il costo atteso è dato da

$$J_\pi(x_0) = E[g_N(x_N) + \sum_{k=0}^{N-1} g_K(x_k, \mu_k(x_k), w_k)]$$

per una legge ottima π* deve quindi valere la relazione $J^*_\pi(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)$ in riferimento allo

3 Il costo viene accumulato nel tempo.

4 In generale l'insieme di vincolo dipende da x_k e dall'indice k.

5 Una legge di controllo per cui vale $\mu_k(x_k) \in U_k(x_k) \forall x_k$ è detta ammissibile (definizione a pag 24).

stato iniziale x_0 .

Tipicamente tramite programmazione dinamica è possibile trovare una scelta ottima π* per tutti gli stati iniziali che risulta perciò indipendente da x_0.

Tramite strategia *closed-loop* è possibile trarre vantaggio da informazione aggiuntiva (valore dello stato corrente) per raggiungere un basso livello di costo[6].

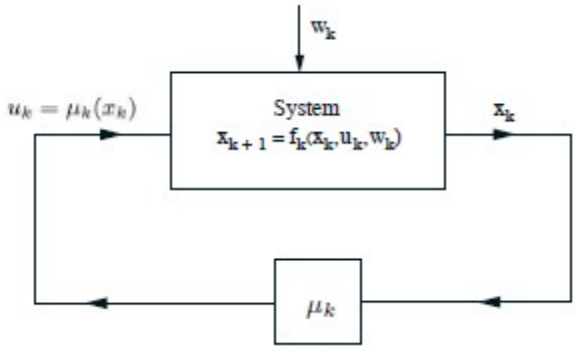

Figura 1.1: *gestione dell'informazione nel problema di base. Il controllore osserva lo stato corrente x_k a applica il controllo u_k dipendente dallo stato.*

Esempio (uso dell'informazione acquisita per la strategia negli scacchi)

Si intende massimizzare la possibilità di vincita di un giocatore in un incontro di due partite fra due concorrenti, i possibili risultati per ogni partita sono:

a)vince uno dei due giocatori (1 punto al vincitore, 0 punti al concorrente)

b)pareggio (½ punto per entrambi i giocatori)

vi sono due possibili scelte di gioco:

- *timid play* (probabilità pd di pareggio e 1-pd di sconfitta)
- *bold play* (probabilità pw di vincita e 1-pw di sconfitta)

In caso di pareggio al secondo turno il giocatore deve adottare la strategia *bold play* per vincere[7].

In applicazione del metodo *closed-loop* supponiamo che un giocatore scelga il primo tipo di strategia solo nel caso in cui risulti in testa nel gioco[8], quindi dopo il primo turno nel quale viene adottata la strategia *bold play* il punteggio è 1-0 per il primo giocatore con probabilità p_w e 0-1 con probabilità $1-p_w$ mentre nel secondo turno viene adottata la scelta *timid play* solo nel primo caso (punteggio 1-0).

Per i due turni la probabilità di vincita risulta $p_w * p_d$, mentre si perderà il gioco con probabilità

6 La riduzione in termini di costo è il "valore dell'informazione".

7 Vedi anche pag 12.

8 La scelta risulterà ottimale assumendo $p_d > p_w$.

$(1-p_w)^2$ e si avrà pareggio con $p_w(1-p_d)+(1-p_w)p_w$.

La prob. di vincere il gioco risulta in definitiva $p_w^2(2-p_w)+p_w(1-p_w)p_d$. Ipotizzando ora che $p_w<1/2$, nessuna strategia *open-loop* può portare a probabilità di vincita superiori al 50% ma, applicando la modalità *closed-loop* sopra indicata, aumentano le possibilità di vittoria con *pw* prossimo a ½ e p_d prossimo a 1 (ad esempio $p_w=0.45$ e $p_d=0.9$ portano a probabilità di vittoria pari a 0.53 circa).

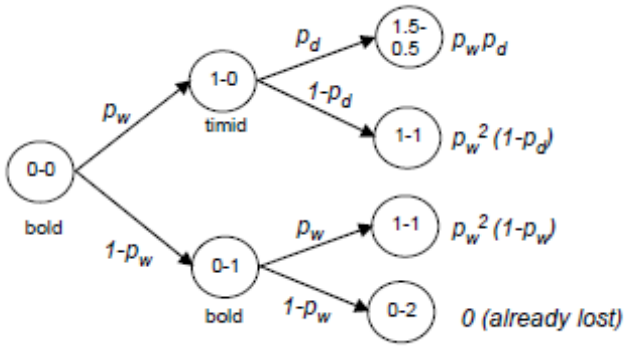

Figura 1.2 *Strategia closed-loop: il giocatore adotta la strategia timid-play se e solo se ha maturato un vantaggio in termini di punteggio per ottenere probabilità di vittoria superiore al 50%.*
I valori a destra vengono accumulati dalla radice alle foglie.

Per rendere evidente il vantaggio che si ottiene con l'uso della tecnica esposta consideriamo la strategia *open-loop* (il giocatore decide la strategia di gioco senza attendere il risultato della prima partita), dato che vi sono due tipi di scelta le possibilità sono 4:

1) strategia *timid play* per entrambe le partite (vittoria con $p_d^2 * p_w$)
2) strategia *bold-play* per entrambe le partite (vittoria con $p_w^2+2p_w^2(1-p_w)=p_w^2(3-2p_w)$)
3) strategia *bold-play* per la prima partita e *timid-play* per la seconda (vittoria con $p_w * p_d+p_w^2(1-p_d)$)
4) strategia *timid-play* per la prima partita e *bold-play* per la seconda (vittoria con $p_w * p_d+p_w^2(1-p_d)$)

La probabilità di vincere il gioco risulta in questo caso *max*($p_w^2(3-2p_w)$, $p_w * p_d+p_w^2(1-p_d)$), con $p_w=0.45$ e $p_d=0.9$ si ottiene prob di vittoria pari a 0.425 circa e il valore di informazione sarà la differenza tra i risultati forniti dalle due tecniche quindi 0.105

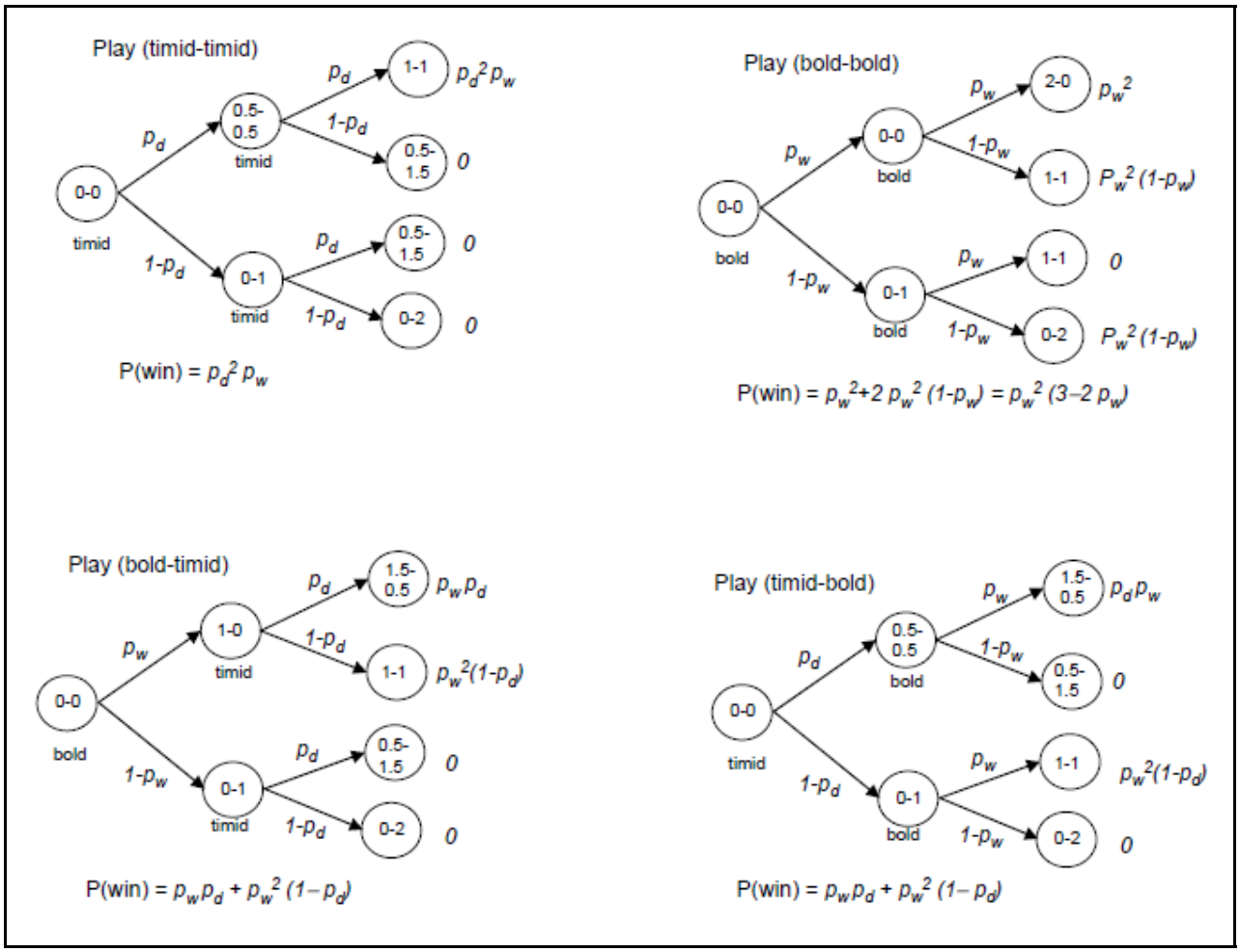

***Figura 1.3**: strategie open-loop.*

L'esempio dimostra come dalla strategia *closed-loop* sia possibile ricavare un vantaggio (seppur minimo) in termini di ottimizzazione nei problemi di tipo non deterministico[9] , la tecnica di programmazione dinamica trae spunto da questa osservazione.

9 Nei problemi deterministici le strategie *open* e *closed loop* sono paragonabili poiché dato un istante k e lo stato corrispondente x_k , il costo atteso dipende esclusivamente da x_k e non da istanti precedenti.

1.2 Principio di ottimalità e algoritmo di programmazione dinamica

L'idea alla base della tecnica di programmazione dinamica(DP) è il principio di ottimalità formulato da R. Bellman (Dynamic Programming, 1957):

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

Riformulando con la notazione in uso:

Principio di ottimalità

Ipotizziamo $\pi^* = (\mu_0^*, \mu_1^*, ..., \mu_{N-1}^*)$ essere una **scelta ottima per il problema di base** e assumiamo che, una volta selezionato π^* vi sia uno stato x_i di riferimento con probabilità positiva. Considerando il sottoproblema per cui si ha x_i al tempo *i* e si vuole minimizzare il costo valido dall'istante *i* all'istante *N*

$$E[g_N(x_N) + \sum_{k=i}^{N-1} g_K(x_k, \mu_k(x_k), w_k)] \ ,$$

Si ha che **la scelta** $(\mu_i^*, \mu_{i+1}^*, ..., \mu_{N-i}^*)$ **risulta ottima per il sottoproblema dato** .

Il principio è di validità generale e può essere verificato tramite la seguente osservazione:

Se il percorso ottimo da un nodo (A) ad un altro nodo (B) attraversa un nodo intermedio (C) , la parte di percorso dal nodo C in avanti è anche il percorso ottimo dal nodo C al nodo B.

Per dimostrare la proposizione si considera il percorso ottimo ***P*** da A a B diviso dal nodo C in due parti (***P₁*** e ***P₂*** **).** Il principio afferma che il percorso ottimo da C a B è ***P₂*** e tale fatto si può provare procedendo per assurdo. Se il percorso ottimo da C a B fosse ***P'₂*** $\neq$***P₂*** , si avrebbe ***costo(P'₂) <costo(P₂).*** Tuttavia si può raggiungere B da A usando ***P₁*** per raggiungere C e poi ***P'₂*** per raggiungere B. Sfruttando l'additività del costo, il costo del percorso congiunto sarà

$$costo(P_1) + costo(P'_2) < costo(P_1) + costo(P_2) = costo(P)$$

questo contraddice l'ipotesi che ***P*** sia il percorso ottimo da A a B.

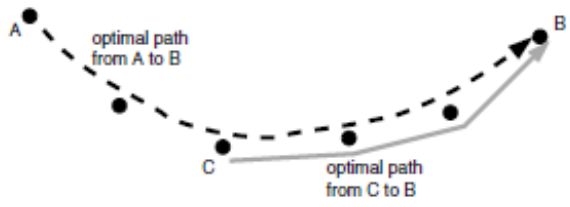

Figura 1.4: *Applicazione del principio di ottimalità. C è nodo attraversato dal percorso ottimo tra A e B, la porzione di percorso da C a B è anche il percorso ottimo da C a B.*

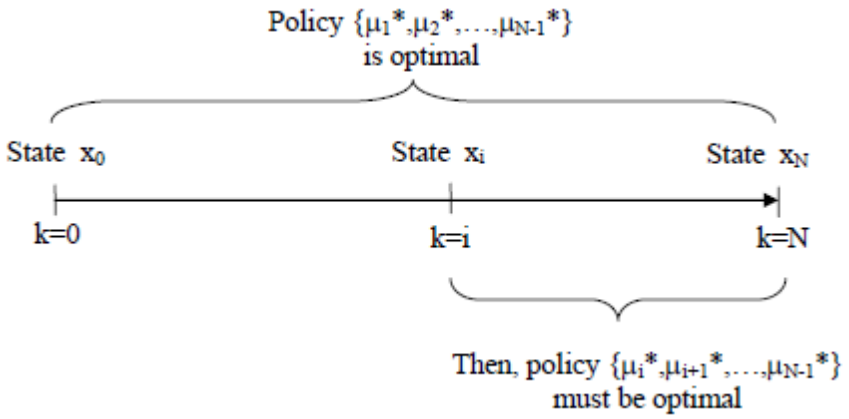

Figura 1.5 *schema del principio di ottimalità.*

Per il principio di ottimalità è possibile trovare una strategia ottima per un determinato problema risolvendo prima i sottoproblemi riferiti agli stati finali del modello corrispondente risalendo alla risoluzione del problema originario. L'algoritmo di programmazione dinamica (DP) procede in maniera sequenziale risolvendo tutti i sottoproblemi di una data lunghezza temporale e usando la soluzione dei sottoproblemi di lunghezza minore.

Algoritmo DP

Proposizione: Per ogni stato iniziale X_0 , il costo ottimo $J^*(x_0)$ del problema base è uguale a $J_0(x_0)$, dato dal seguente algoritmo che procede dall'istante *N-1* all'istante *0:*

$$J_N(x_N) = g_N(x_N) \quad ,$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k}[g_k(x_k,u_k,w_k)+J_{k+1}(f_k(x_k,u_k,w_k))], \quad k=0,1,\ldots,N-1.$$

$J_0(x_0)$, generato al passo finale, è uguale al costo ottimo $J^*(x_0)$.

La scelta $\pi^*=(\mu_0^*, \mu_1^*, \ldots\ldots, \mu_{N-1}^*)$ per cui $\mu_k^*(x_k)=u_k^*$ rende minimo il secondo membro per ogni x_k ed ogni indice *k*, è anch'essa ottima.

Dimostrazione[10]

Per induzione dimostrando che $J_k(x_k)$ è uguale a $J^*_k(x_k)$, definito come costo ottimo del sottoproblema finale relativo allo stato x_k e al tempo *k.* Tutti i sottoproblemi vengono risolti assieme al problema originale, ma al prezzo di elevate richieste computazionali.

Poniamo $\pi^k = (\mu_k, \mu_{k+1}, \ldots\ldots, \mu_{N-1})$ come scelta per il sottoproblema finale da istante k in avanti, per k=N definiamo $J_N^*(x_N) = J_N(x_N) = g_N(x_N)$.

Assumiamo $J_{k+1}(x_{k+1}) = J_{k+1}^*(x_{k+1})$. Allora, con $\pi^k = (\mu_k, \pi^{k+1})$

10 Ipotizziamo che w_k e le quantità coinvolte dal valore atteso assumano un numero di valori finito. La dimostrazione rigorosa non tiene conto di tali ipotesi.

$$J_k^*(x_k) = \min_{\mu_k, \pi^{k+1}} \underset{w_k, \ldots, w_{N-1}}{E} [g_k(x_k, \mu_k(x_k), w_k) + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_i)]$$

$$= \min_{\mu_k} \underset{w_k}{E} [g_k(x_k, \mu_k(x_k), w_k) + \min_{\pi^{k+1}} \underset{w_{k+1}, \ldots, w_{N-1}}{E} [g_N(x_N) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_i)]]$$

$$= \min_{\mu_k} \underset{w_k}{E} [g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^*(f_k(x_k, \mu_k(x_k), w_k))]$$

$$= \min_{\mu_k} \underset{w_k}{E} [g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}(f_k(x_k, \mu_k(x_k), w_k))] \text{ (ip. induttiva)}$$

$$= \min_{u_k \in U_k(x_k)} \underset{w_k}{E} [g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))] = J_k(x_k)$$

Nel secondo passaggio si sfrutta il principio di ottimalità, nella quarta equazione si fa uso dell'ipotesi induttiva mentre nell'ultimo passaggio si converte la minimizzazione su μ_k in minimizzazione su u_k (per ogni funzione F di x e u si ha $\min_{\mu \in M} F(x, \mu(x)) = \min_{u \in U(x)} F(x, u)$ dove M è l'insieme di funzioni $\mu(x)$ tali che $\mu(x) \in U(x)$ per ogni x. Q.E.D.

In molti casi una soluzione analitica non è possibile e si ricorre a metodi numerici per i quali il metodo DP può divenire proibitivo dal punto di vista del costo computazionale ma rappresenta un procedimento base per altri approcci subottimali.[11]

Esempio (ottimizzazione per la strategia negli scacchi)

In riferimento all'esempio precedente lo scopo è quello di formulare un algoritmo DP per trovare la scelta che massimizza la probabilità del giocatore di vincere l'incontro; possiamo adattare l'algoritmo DP al problema di massimizzazione.

Consideriamo il caso di un incontro con N partite e poniamo che lo stato rappresenti la differenza tra i punti del giocatore e quelli dell'avversario (0 corrisponde a parità). La funzione di costo ottimo alla k-esima partita viene formulata ricorsivamente tramite programmazione dinamica:

$$J_k(x_k) = max[P_d * J_{k+1}(x_k) + (1 - P_d) J_{k+1}(x_k - 1), P_w * J_{k+1}(x_k + 1) + (1 - P_w) J_{k+1}(x_k - 1)] \quad (*)$$

per la ricerca del massimo si osservano le due possibili scelte:

(a) *timid play* che mantiene il punteggio a x_k con probabilità P_d e porta a x_{k-1} con probabilità 1-P_d.

(b) *bold play* che porta il punteggio a x_{k+1} o x_{k-1} con probabilità P_w *o* 1-P_w rispettivamente.

Conviene scegliere la strategia *bold play* se si verifica:

$$P_w * J_{k+1}(x_k + 1) + (1 - P_w) J_{k+1}(x_k - 1) \geq P_d * J_{k+1}(x_k) + (1 - P_d) J_{k+1}(x_k - 1)$$

o in maniera equivalente $\frac{P_w}{P_d} \geq \frac{J_{k+1}(x_k) - J_{k+1}(x_k - 1)}{J_{k+1}(x_k + 1) - J_{k+1}(x_k - 1)}$

11 Riferimento a capitolo 6 del testo *Dynamic programming and Optimal Control [1]*.

il passo iniziale della ricorsione è dato da $J_N(x_N)=\begin{Bmatrix}1\ se\ x_N & >0\\ P_w\ se\ x_N & =0\\ 0\ se\ x_N & <0\end{Bmatrix}$. Eseguendo l'algoritmo DP (*) tenendo conto delle due condizioni sopra valide per la strategia ottima *bold play* e assumendo $P_d > P_w$ si ottiene

$J_{N-1}(x_N-1) = 1 \quad per\ x_{N-1}>1;$ *scelta ottima*: *entrambe*

$J_{N-1}(1) = max[P_d+(1-P_d)*P_w, P_w+(1-P_w)*P_w]$

$= P_d+(1-P_d)*P_w$ *scelta ottima*: *timid play*

$J_{N-1}(0) = P_w$ *scelta ottima*: bold *play*

$J_{N-1}(-1) = P_w^2$ *scelta ottima*: bold *play*

$J_{N-1}(x_N-1) = 0 \quad per\ x_{N-1}<-1;$ *scelta ottima*: *entrambe*

data $J_{N-1}(x_N-1)$ e le condizioni sopra si conclude

$J_{N-2}(0) = max[P_d*P_w+(1-P_d)*P_w^2, P_w*(P_d+(1-P_d)*P_w)+(1-P_w)*P_w^2] =$

$= P_w(P_w+(P_w+P_d)(1-P_w))$.

Se si ha punteggio di parità con due partite rimaste da giocare, *bold play* è la strategia ottima. Come osservato in precedenza per un incontro di due partite la scelta ottima è *timid play* se e solo se il giocatore risulta in vantaggio al primo turno.

CAPITOLO 2
IL PROBLEMA DEL PERCORSO MINIMO

2.1 Sistemi deterministici e percorso minimo

L'attenzione si sposta ora sui problemi deterministici cioè problemi nei quali si assume che ogni disturbo w_k possa assumere un solo valore. In questo contesto i risultati acquisiti nei processi successivi (forniti dal meccanismo di *feedback*) non portano a vantaggi in termini di riduzione dei costi perché, data una scelta $(\mu_0, \mu_{1,\dots}, \mu_{N-1})$ e lo stato iniziale X_0, si possono prevedere gli stati futuri tramite $x_{k+1} = f_k(x_k, \mu_k(x_k)), \qquad k = 0,1,\dots,N-1$
mentre il controllo è prevedibile come $u_k = \mu_k(x_k), \qquad k = 0,1,\dots,N-1$.

Prendiamo in considerazione un problema dove lo spazio di stato sia un insieme finito per ogni k, possiamo associare un controllo u_k ad ogni stato $f_k(x_k, u_k)$ con costo $g_k(x_k, u_k)$ tramite una transizione. È possibile quindi rappresentare un problema deterministico a stati finiti tramite un grafo dove gli stati corrispondono ai nodi, le variabili di controllo agli archi e le sequenze di controllo (*open-loop*) ai percorsi da stato iniziale a stato finale.

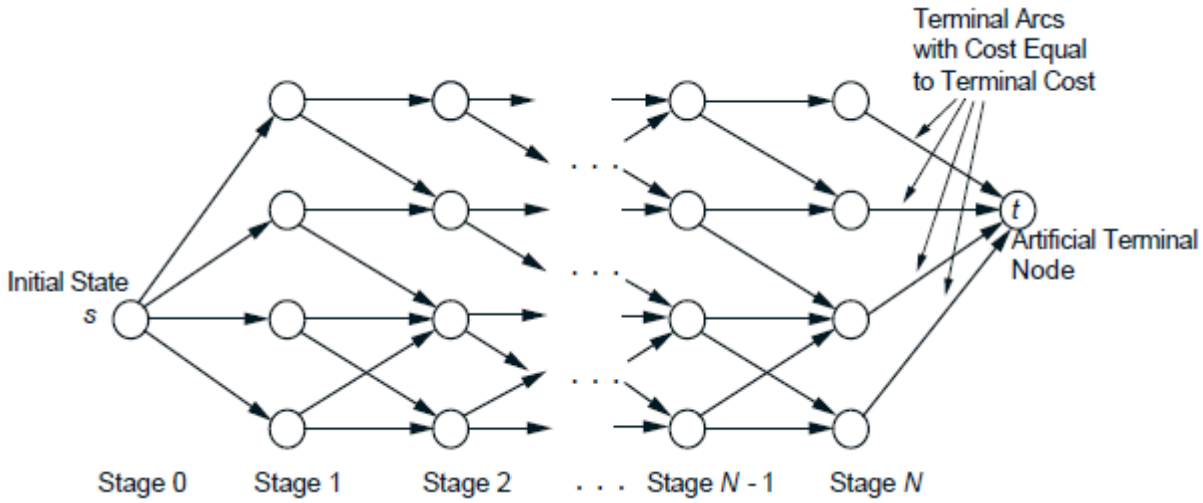

Figura 2.1*: grafo di transizione per un sistema a stati finiti deterministico. La lunghezza di un arco rappresenta il costo di una transizione e il problema viene ricondotto al calcolo del percorso minimo dal nodo iniziale s al nodo finale t.*

Indichiamo con a_{ij}^k il costo della transizione dallo stato $i \in S_k$ allo stato $j \in S_{k+1}$ al tempo k, a_{it}^N è il costo finale dello stato $i \in S_N$.

Formuliamo l'algoritmo DP per il problema del percorso minimo considerando il procedimento in due varianti (*DP backward e DP forward*).

Algoritmo DP (backward)

$$J_N(i) = a_{it}^N, \qquad i \in S_N$$
$$J_k(i) = \min_{j \in S_{k+1}} [a_{ij}^k + J_{k+1}(j)], \quad i \in S_k, \qquad k = 0,1,\ldots,N-1.$$

il costo ottimo è $J_0(s)$ ed è uguale alla lunghezza del percorso minimo da *s* a *t*.

Osserviamo che un percorso ottimo di tipo $s \to t$ è anche un percorso ottimo di tipo $t \to s$ relativo al problema inverso dove la direzione di ogni arco viene invertita mentre la lunghezza del percorso rimane invariata. Tale osservazione permette una riformulazione dell'algoritmo DP.

Algoritmo DP (forward)

$$\tilde{J}_N(j) = a_{sj}^0, \qquad j \in S_1$$
$$\tilde{J}_k(j) = \min_{i \in S_{N-k}} [a_{ij}^{N-k} + \tilde{J}_{k+1}(i)], \quad j \in S_{N-k+1}.$$

il costo ottimo è $\tilde{J}_0(t) = \min_{i \in S_N} [a_{it}^N + \tilde{J}_1(i)]$, mentre $\tilde{J}_k(j)$ indica il costo ottimo da *s* a *j*.

Le due versioni conducono allo stesso risultato per $J_0(s) = \tilde{J}_0(t)$ e la tipologia *DP forward* è resa possibile grazie alla formulazione in termini di percorso minimo nel contesto dei sistemi deterministici. Dal momento che, ogni problema di percorso minimo può essere posto come problema DP a stati finiti deterministico, problemi generici di percorso minimo possono essere strutturati nella seguente modalità:

Data la sequenza {1,2....., N, t} indicante i nodi del grafo (t = nodo destinazione)e indicando con a_{ij} *il costo per il percorso dal nodo i al nodo j, trovare il percorso minimo (a minimo costo) da ogni nodo i al nodo t. Assumendo che tutti i cicli del grafo abbiano lunghezza non negativa, la ricerca di un percorso ottimo non richiede più di N passi (N = numero di nodi). Assumiamo che il problema richieda N passi e tolleriamo percorsi degeneri da un nodo i a se stesso con costo* $a_{ii}=0$.

$J_k(i)$ *sarà il costo ottimo da i a t in N-k passi,* $J_0(i)$ *il costo del percorso ottimo da i a t.*

Algoritmo DP: $J_k(i) = \min_{j=1,\ldots,N} [a_{ij} + J_{k+1}(j)], \quad k = 0,1,\ldots,N-2 \quad , \quad J_{N-1}(i) = a_{it}$

2.2 Applicazioni e algoritmi di percorso minimo

Il problema del percorso minimo appare in molti contesti e l'insieme di possibili applicazioni è piuttosto vasto. In questa sezione vengono discussi alcuni esempi rappresentativi e algoritmi di possibile utilizzo.

Esempio (Project Management)

Si consideri la pianificazione di un progetto che coinvolge molte attività, alcune delle quali devono necessariamente terminare prima che altre abbiano inizio. Si conosce la durata di ogni attività e si vuole trovare la durata di tempo totale richiesta per portare a termine il progetto stabilendo quindi le *attività critiche* dalle quali dipende la durata dell'intero lavoro.
Si può rappresentare il problema tramite un grafo in cui i nodi riassumono le fasi di progetto e gli archi (i,j) le attività intraprese da una fase all'altra con durata $t_{ij}>0$. La rete delle attività è aciclica e per ogni percorso $p = \{(1,j_1)....(j_k,i)\}$ dal nodo 1 al nodo i sia D_p la somma delle durate delle varie attività è $D_p = t_{1,j_1} + t_{j_1,j_2} + \ldots + t_{j_k,i}$. Per la fase *i* il tempo richiesto sarà dato da $T_i = \max_p [D_p]$

.

Dal momento che la rete è aciclica, esisterà un numero finito di percorsi da *1* a *p* per cui è possibile raggiungere il valore massimo. Per trovare T si cerca quindi il percorso più lungo da 1 ad i (oppure il percorso minimo con la lunghezza di ogni arco (i,j) pari a $-t_{ij.}$)[12]; con il percorso minimo da 1 a N si determina allora la durata del progetto.

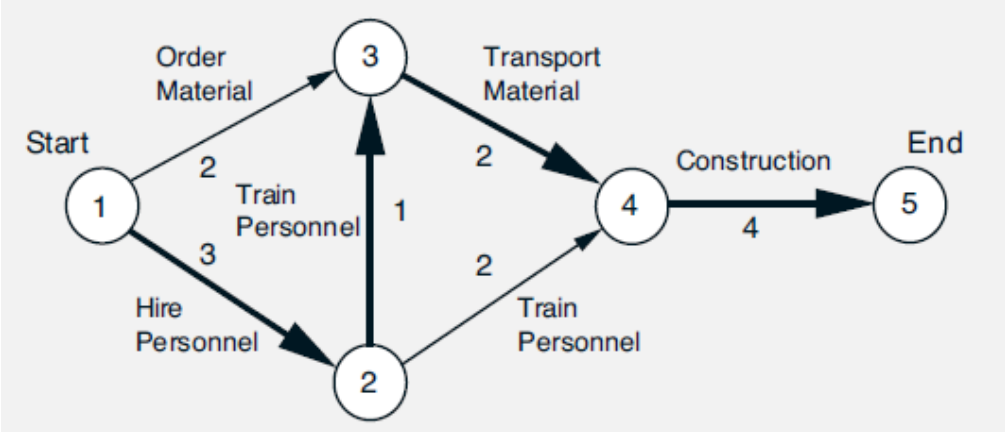

Figura 2.2: *esempio di una rete di attività. Una fase viene completata se tutte le attività associate agli archi entranti nel nodo corrispondente sono completate.*
Il percorso critico viene evidenziato in grassetto, se una qualsiasi attività del percorso critico subisce un ritardo, l'intero progetto subisce un aumento di durata.

12 L'ipotesi di grafo aciclico giustifica questa possibilità.

Denotando con S_k l'insieme di fasi che non dipendono dal completamento di altre, si può vedere l'insieme come lo spazio di stato del problema DP equivalente.

Ricercando la massimizzazione e cambiando segno alle lunghezze degli archi, scriviamo l'algoritmo

$$T_i = \max_{(j,i)\ t.c.\ j \in S_{k-1}} [t_{ji} + T_j], \qquad per\ ogni\ \ i \in S_k\ \ con\ \ i \notin S_{k-1}$$

che figura nella variante *DP forward* (dall'origine 1 a destinazione N).

Osservando il grafo in esempio si ha

$$S_0 = \{1\}\ ,\ S_1 = \{1,2\}\ ,\ S_2 = \{1,2,3\}\ ,\ S_3 = \{1,2,3,4\}\ ,\ S_4 = \{1,2,3,4,5\}$$

$$T_1 = 0\ ,\ T_2 = 3\ ,\ T_3 = 4\ ,\ T_4 = 6\ ,\ T_5 = 10\ .$$

Il percorso critico risulta $1 \to 2 \to 3 \to 4 \to 5$

.Esempio (Algoritmo di Viterbi)

L'algoritmo di Viterbi sfrutta la tecnica di programmazione dinamica per trovare una sequenza di stati nei problemi che coinvolgono modelli a catene di Markov nascoste[13]. L'algoritmo venne concepito da Andrew Viterbi nel 1966 come algoritmo di decodifica per codici convoluzionali[14] ma per la sua generalità è possibile adeguarlo alla descrizione di fenomeni di diverso genere.

Per introdurre il procedimento consideriamo come modello una catena di Markov con probabilità di transizione p_{ij} e stati corrispondenti alle transizioni nascosti, ogni transizione è nota per osservazioni indipendenti.

Data una sequenza di osservazioni si vuole determinare la sequenza di transizioni corrispondenti. Sia *r(z; i,j)* la probabilità che l'osservazione assuma valore z con transizione di stato da i a j. Nota la sequenza $Z_N = \{z_1, z_2, \ldots, z_N\}$ vogliamo trovare la sequenza di transizioni più probabile $\hat{X}_N = \{\hat{x}_0, \hat{x}_{1,\ldots}, \hat{x}_N\}$ che massimizza $p(X_N|Z_N)$ su ogni $X_N = \{x_0, x_{1,\ldots}, x_N\}$.

Per la formulazione come problema di percorso minimo si usa un particolare diagramma di coppie stato-tempo (*trellis diagram* o diagramma a traliccio) concatenando le N+1 coppie dello spazio di stato precedute e seguite dai nodi s e t.

13 Un modello di Markov nascosto (*Hidden Markov Model* - HMM) è una catena di Markov i cui stati non sono osservabili direttamente: gli stati evolvono secondo una catena di Markov, ogni stato genera un evento con una certa distribuzione di probabilità che dipende solo dallo stato, solo l'evento è osservabile.

14 Tipo particolare di codice per la correzione d'errore in Telecomunicazioni.

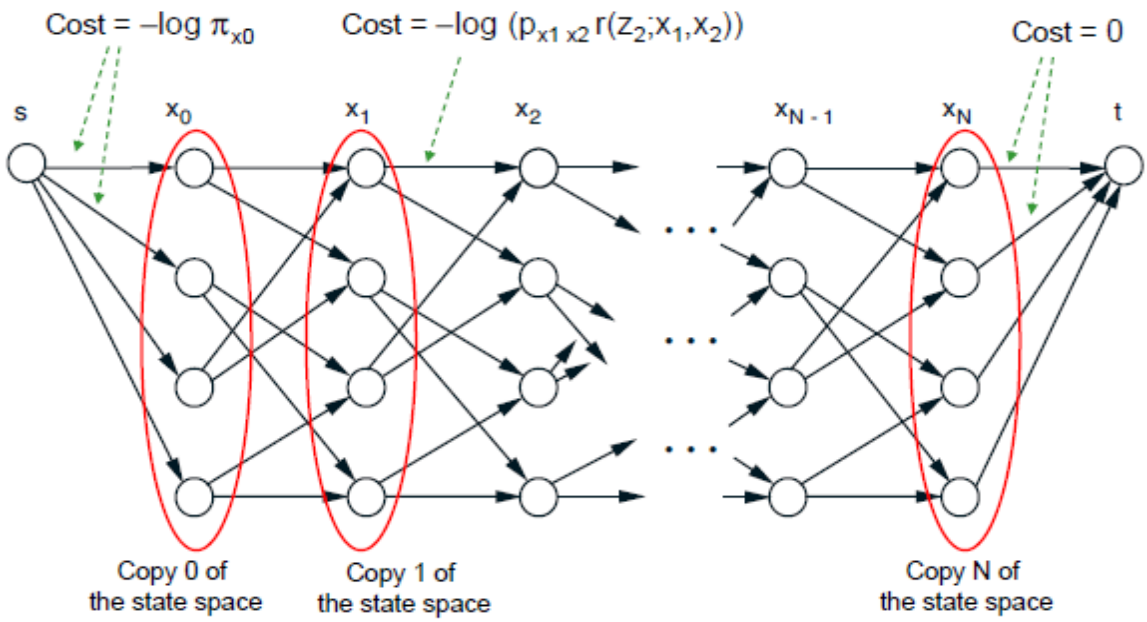

Figura 2.3: *esempio di costruzione di diagramma a traliccio. La figura rappresenta la stima di stato per una catena di Markov a stati nascosti (HMM) come problema di percorso minimo da s a t. Un arco connette x_{k-1} con x_k se $p_{xk-1,xk}>0$.*

$\hat{X}_N$ può essere trovato risolvendo un problema di percorso minimo con grafo aciclico.

Avendo $P(X_N|Z_N) = \frac{P(X_N, Z_N)}{P(Z_N)}$, massimizzare il primo membro è equivalente a massimizzare il logaritmo $ln(P(X_N,Z_N))$. Per quanto detto

$$P(X_N, Z_N) = \pi_{x_0} \prod_{k=1}^{N} p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k) \quad ,[15]$$

quindi il problema è equivalente a minimizzare $-\ln(\pi_{x_0}) - \sum_{k=1}^{N} \ln(p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k))$ su tutte le possibili sequenze $\{x_0,x_1,...,x_N\}$. Il problema di minimizzazione è equivalente al problema del percorso minimo dal nodo s al nodo t nel diagramma e il percorso è definito dalla sequenza di stati $\{\hat{x}_0, \hat{x}_1, ..., \hat{x}_N\}$. Il percorso minimo viene calcolato in maniera sequenziale tramite la variante *DP forward* inizialmente dal nodo s al nodo x_1, poi usando le distanze trovate per calcolare il percorso minimo da s ad ogni nodo x_2 ecc.

Formuliamo quindi il passo ricorsivo DP:

$$\boldsymbol{D_{k+1}(x_{k+1}) = \min_{x_k \, t.c. \, p_{x_k x_{k+1}} > 0} [D_k(x_k) - \ln(p_{(x_k x_{k+1})} r(z_{k+1}; x_k, x_{k+1}))]}$$

con condizione iniziale $\boldsymbol{D_0(x_0) = -\ln(\pi_{x_0})}$.

Un vantaggio di questa procedura è che può essere eseguita in tempo reale ogni qual volta si ottiene una nuova osservazione. È possibile stimare una porzione della sequenza di stato senza ricevere

15 Vengono omessi i passaggi dove si usano proprietà della catena di Markov e l'indipendenza degli eventi.

l'intera sequenza di osservazione Z_N , ad esempio per un dato indice k si possono trovare tutti i percorsi minimi da s a X_k passanti per un singolo nodo nel sottografo $x_0,x_1,...,x_{k-1}$, per la determinazione del minimo percorso da s a quel nodo si procede alla stima della sottosequenza dal nodo in avanti senza bisogno di osservazioni aggiuntive. La procedura di stima sopra elencata descrive a livello intuitivo l'algoritmo di Viterbi.

Vista l'equivalenza tra problemi di percorso minimo e problemi di controllo ottimo per sistemi a stati finiti deterministici, la tecnica DP può essere usata per risolvere problemi di percorso minimo. Esistono algoritmi ad elevate prestazioni per lo stesso problema ma il procedimento DP viene scelto in tutti i casi in cui si ha la struttura di grafo aciclico e vi sia disponibilità di calcolo parallelo[16]. Tecniche diverse dalla programmazione dinamica sono preferibili se evitano il calcolo del costo ottimo per ogni stato[17] che in reti molto estese aumenta la complessità in maniera considerevole.

Di seguito verranno esposti algoritmi generali per il problema di percorso minimo e i loro legami con la tecnica DP.

Metodi *label correcting* e *label setting*

Per l'utilizzo dei metodi a "correzione di etichetta" (***label correcting***) vengono forniti i nodi iniziale e finale (s e t) con le lunghezze $a_{ij} > 0$, l'idea è quella di scoprire in maniera progressiva i percorsi minimi dall'origine s ad ogni altro nodo i. Per descrivere il procedimento generale si usa la seguente notazione:

- d_i (etichetta di i) è la lunghezza del percorso minimo trovato (inizialmente $d_s = 0$, $d_i = \infty$ per ogni $i \neq s$)
- UPPER: etichetta per il nodo destinazione
- OPEN list : contiene nodi "attivi" cioè candidati per l'analisi. (inizialmente OPEN = {s})

Il procedimento viene quindi descritto come *label correcring Algorithm (LCA)*

Step 1 (Node Removal): Remove a node i from OPEN and for each child j of i, do step 2

Step 2 (Node Insertion Test): If $d_i + a_{ij} < \min\{d_j, \text{UPPER}\}$, set $d_j = d_i + a_{ij}$ and set i to be the parent of j. In addition, if $j \neq t$, place j in OPEN if it is not already in OPEN, while if $j = t$, set UPPER to the new value $d_i + a_{it}$ of d_t

Step 3 (Termination Test): If OPEN is empty, terminate; else go to step 1

16 Esecuzione simultanea su più microprocessori per ottenere aumento di prestazione.
17 Nella tecnica DP ogni nodo ed ogni arco viene coinvolto nel calcolo.

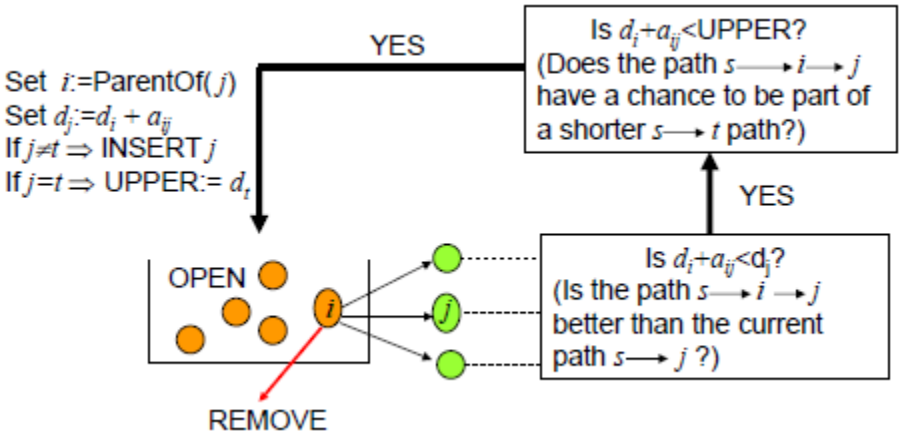

Figura 2.4: *algoritmo label correcting (test di inserzione di un nodo nella lista OPEN).*

La validità dell'algoritmo generale viene stabilita dalla

Proposizione: se esiste almeno un percorso dall'origine alla destinazione, l'algoritmo *label correcting* termina con UPPER uguale alla distanza minima dall'origine alla destinazione.

Dimostrazione:

(1) Ogni volta che un nodo j entra in OPEN, l'etichetta viene diminuita e diviene uguale alla lunghezza di un percorso da s a j.

(2) Il numero di possibili lunghezze di percorso distinte è finito quindi il numero di volte che un nodo può entrare in OPEN è finito e l'algoritmo termina.

(3) Poniamo $(s,j_1, j_2,....,j_k,t)$ come percorso minimo e d* come distanza minima. Se UPPER>d* al termine, UPPER sarà maggiore della lunghezza di tutti i percorsi $(s,j_1, j_2,....,j_m)$ m = 1,...,k. Quindi j_k non entrerà nella lista OPEN con d_{jk} uguale alla distanza minima da s a j_k . Analogamente il nodo j_{k-1} non entrerà nella lista OPEN con dj_{k-1} uguale alla distanza minima da s a j_{k-1} . Si procede quindi fino a j_1 per ottenere una contraddizione, segue che, al termine, UPPER sarà uguale alla distanza minima da s a t. Q.E.D

Per rendere il metodo efficiente (ridurre la complessità) si possono adottare diverse strategie :

- ridurre il valore di UPPER il più velocemente possibile(scoprire percorsi da s a t in velocità)
- tenere basso il numero di entrate in OPEN (rimuovere inizialmente da OPEN i nodi con etichette di basso valore e usare euristica : se d_i è basso allora d_j una volta impostato a $d_i + a_{ij}$ assumerà valore basso, quindi vi è bassa probabilità di entrata di j nella lista OPEN.
- Ridurre in numero di nodi selezionati per la rimozioni da OPEN

Considerando la selezione del nodo da rimuovere dalla lista OPEN, varie possono essere le scelte, tra le più comuni vi sono le procedure presenti negli algoritmi di Dijkstra e Bellman-Ford. Di seguito vengono esposti i procedimenti con cenni alle relative complessità algoritmiche.

Breadth-first search (metodo Bellman-Ford)

Si sfrutta la scelta first-in/first-out, il nodo viene sempre rimosso dalla cima della lista OPEN[18] e ogni nodo entrante in OPEN viene posizionato alla fine della lista.

Il principio di ottimalità precedentemente esposto porta alla definizione dell'equazione ricorsiva nota come equazione di Bellman nelle variabili $V_1,....,V_N$ per le lunghezze dei cammini

$$V_s = 0, \qquad V_j = \min_{(ij)\in E}[V_i + c_{ij}] \qquad \forall\, j \neq s$$

In base al principio di ottimalità i valori ottimi sono soluzioni dell'equazione di Bellman.

Il modello presentato considera cammini uscenti da un nodo prefissato s e cammini entranti in un nodo t; se P è un cammino generico con destinazione in t e j è un nodo di P indichiamo con P^j la restrizione da j a t di P. La lunghezza di un cammino P può essere ricorsivamente definita (da t operando a ritroso) come

$$V(P^t) = 0, \qquad V(P^i) = V(P^j) + c_{ij} \qquad \forall (ij) \in P$$

L'equazione di Bellman nelle variabili $V^1,....,V^N$ (per problemi di minimo) diventa

$$V^t = 0, \qquad V^i = \min_{j:ij\in E}[V^j + c_{ij}] \qquad \forall\, i \neq t \quad ,$$

L'algoritmo risolutivo, noto come algoritmo di Bellman-Ford itera, usando valori provvisori a partire da Vs = 0, Vi = ∞, i≠s il calcolo $V_j = min[V_j ; V_i + c_{ij}] \qquad \forall (ij) \in E$. *E = insieme archi*

Algoritmo di Bellman-Ford

```
(*versione in avanti*)
input(G, c)
    V_s := 0; for all i ≠ s do V_i := ∞; p(s) := s;
    for k := 1 to n do
        for (ij) ∈ E do
            if V_j > V_i + c_ij
            then begin
            V_j := V_i + c_ij;
            p(j) := i;
            end
output(V, p).
```

```
(*versione all'indietro*)
input(G, c)
    V^t := 0; for all i ≠ t do V^i := ∞; p(t) := t;
    for k := 1 to n do
        for (ij) ∈ E do
            if V^i > V^j + c_ij
            then begin
            V^i := V^j + c_ij;
            p(i) := j;
            end
output(V, p).
```

Figura 2.5 *Dato un grafo orientato G=(N,E) con n = |N| e m = |E|, sia s un nodo prefissato di G. L'algoritmo di Bellman-Ford (nelle due versioni) trova i cammini ottimi oppure stabilisce l'assenza di tali cammini con complessità O(nm). Se l'algoritmo termina entro n iterazioni i suoi valori soddisfano l'equazione di Bellman.*

Se i costi sono non negativi e il grafo è generico si può risolvere l'equazione di Bellman in modo meno dispendioso dal punto di vista della complessità usando il procedimento di Dijkstra.

18 Si assume che OPEN sia strutturata come struttura a coda (queue).

Best-first search (Dijkstra)

L'algoritmo di Dijkstra può essere visto come specifico metodo *label correcting* o come esempio del metodo ***label setting***, per il quale un nodo *i* rimosso dalla lista di candidati OPEN presenta etichetta minima ad ogni iterazione. Ogni nodo viene perciò inserito in OPEN al più una volta ed i nodi entrano nella lista con la minima distanza. L'algoritmo viene solitamente riferito alla tipologia *greedy*[19], ma è fortemente ispirato al principio di ottimalità di Bellman (pag 8) e rappresenta la principale procedura di approssimazione successiva per la tecnica DP[20].

Il legame con il metodo di programmazione dinamica è evidente dalla prima formulazione dell'algoritmo (Edsger W. Dijkstra. *A note on two problem in connexion with graphs*. Numerische Mathematik, 269– 271, 1959):

> **Problem 2.** Find the path of minimum total length between two given nodes P and Q.
>
> We use the fact that, if R is a node on the minimal path from P to Q, knowledge of the latter implies the knowledge of the minimal path from P to R.

L'articolo di Dijkstra è riferito a due problematiche nella teoria dei grafi: il problema 1 per la ricerca del minimo albero ricoprente[21] e il problema 2 per la ricerca del percorso minimo.

Per quanto detto l'algoritmo di Dijkstra ,anche se non viene esplicitamente riferito alla tecnica DP, trova ispirazione nel principio fondante della programmazione dinamica che viene sfruttata per ottenere un metodo risolutivo per il percorso minimo efficiente dal punto di vista del costo computazionale.

Algoritmo di Dijkstra
input(G, c);
  $V_s := 0$; **for** all $i \neq s$ **do** $V_i := \infty$;
  $S := \{s\}$; $k := s$; $p(s) = s$;
  repeat
    for $(kj) \in E$, $j \notin S$ **do**
      if $V_j > V_k + c_{kj}$
      then begin
        $V_j := V_k + c_{kj}$;
        $p(j) = k$
      end
    $k := \text{argmin}_{j \notin S} V_j$;
    $S := S \cup \{k\}$;
  until $(S = N) \vee (V_k = \infty)$
output(V, p)

19 Tecnica di ottimizzazione per la scelta localmente ottima in un problema generico a differenza della tecnica DP che assicura la scelta globalmente ottima. I due approcci sono simili ma vengono differenziati per quanto riguarda la gestione dei sottoproblemi.

20 Il metodo di approssimazione successive viene usato dalla tecnica DP per risolvere l'equazione funzionale associata a problemi di percorso minimo. [5]

21 Albero contenente tutti i vertici di un grafo connesso per cui è minima la somma dei "pesi" associati agli archi.

Figura 2.6 *Dati costi non negativi e grafo generico l'algoritmo di Dijkstra risolve il problema del percorso minimo. L'algoritmo implementa il seguente ragionamento induttivo: sia noto durante l'iterazione un insieme S di nodi con valori* V_j , $j \in S$ *e siano noti valori* V_j *con* $j \notin S$ *che rappresentano i costi dei cammini ottimi da s a j con il vincolo di usare come nodi intermedi soltanto i nodi di S (se il cammino non esiste* $V_j = \infty$*). Sia k tale che* $V_k = \min_{j \notin S} V_j$. V_k *è il valore finale ottimo per i cammini da s a k.. Infatti ogni altro cammino dovrebbe passare per qualche altro nodo non in S ad un costo più elevato a causa della scelta di K e delle distanze non negative. Si può aggiornare S ponendo* $S := S \cup \{k\}$. *Per aggiornare* V_j *con* $j \notin S$, *bisogna passare per K, sfruttando il principio di ottimalità si pone quindi* $V_j := min[V_j ; V_k + c_{kj}]$.*La proprietà è ora verificata per un insieme più grande di nodi e si può iterare finché S = N. L'algoritmo di Dijkstra implementa questo ragionamento induttivo ed ha complessità* $O(n^2)$ *o O(m log(n)) a seconda dell'implementazione.*

Il problema di trovare un cammino minimo con costi non negativi si può quindi agevolmente risolvere con la variante di Dijkstra, se i costi possono essere negativi ma non vi sono cicli negativi il problema rimane risolubile con l'algoritmo di Bellman-Ford (ma non con quello di Dijkstra). In presenza di cicli negativi il problema diventa *NP-hard*[22] e vi sono algoritmi euristici risolutivi radicalmente diversi da quelli sopra esposti.

Gli algoritmi di Dijkstra e Bellman-Ford fanno quindi riferimento alla tecnica di ottimizzazione DP e vengono scelti per la soluzione al problema del percorso minimo in vari contesti come ad esempio le strategie di *routing* nell'ambito delle telecomunicazioni.

Per quanto detto il problema del cammino minimo costituisce un semplice ambito applicativo per la teoria del controllo ottimo e gli algoritmi risolutivi efficienti traggono ispirazione dalla programmazione dinamica.

22 Problemi tali per cui un algoritmo risolutivo può essere convertito in un algoritmo per risolvere un qualunque problema NP. Per i problemi NP è noto un algoritmo che termina in tempo polinomiale rispetto alla dimensione dei dati nel caso si possa utilizzare un numero indeterminato di macchine in parallelo.

CAPITOLO 3
CONTROLLO OTTIMO NEL CONTINUO

3.1 Equazione di Hamilton-Jacobi-Bellman

Introduciamo ora un modello per sistemi dinamici a tempo continuo:

$$\dot{x}(t) = f(x(t), u(t)), \qquad 0 \le t \le T, \quad x(0) \ \textit{dato} \quad \text{con}$$

- $\boldsymbol{x(t)} \in \mathbb{R}^n$: vettore di stato al tempo t
- $\boldsymbol{u(t)} \in U \subset \mathbb{R}^m$: vettore controllo al tempo t
- U: insieme dei vincoli di controllore
- T: istante finale

definiamo **traiettorie di controllo ammissibili** le funzioni continue a tratti $\{u(t) | t \in [0,T]\}$ con $u(t) \in U \ \forall\, t \in [0,T]$; per ogni traiettoria di controllo ammissibile assumiamo che il sistema di equazioni differenziali prima decritto abbia soluzione unica $\{x(t) | t \in [0,T]\}$ (traiettoria di stato).

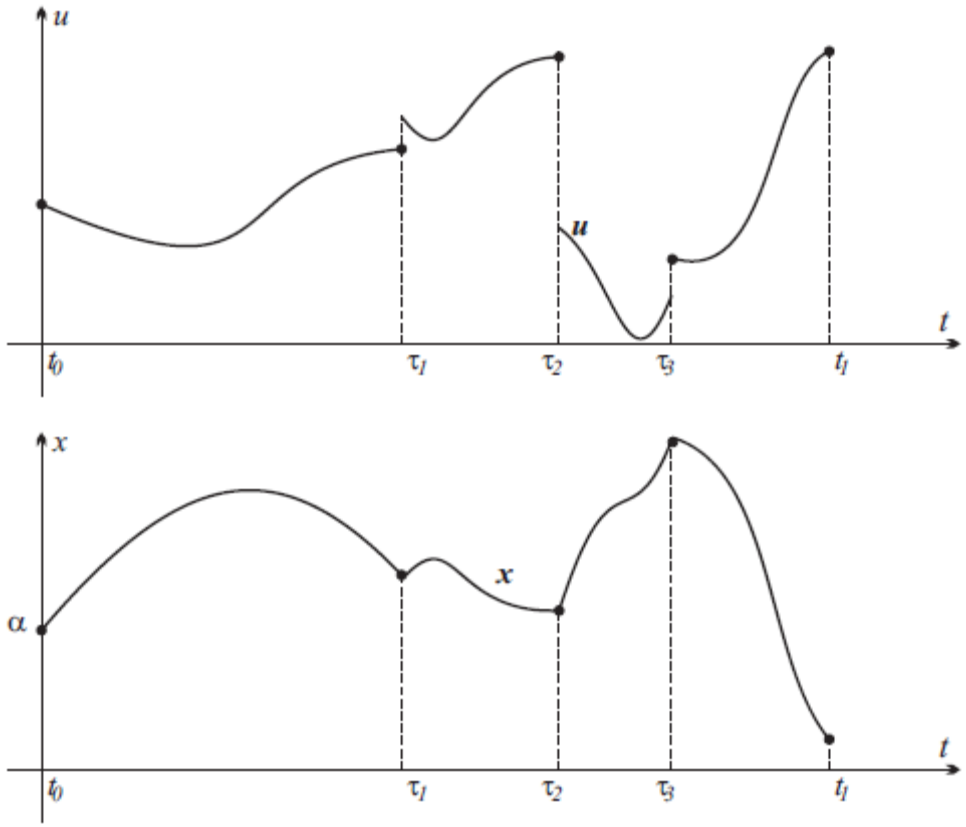

Figura 3.1: *Un controllo ammissibile e la traiettoria corrispondente.*

L'obiettivo è quello di trovare una traiettoria di controllo ammissibile e la traiettoria di stato corrispondente che minimizzano il costo $h(x(T))+\int_0^T g(x(t),u(t))\,dt$, f,g,h si assumono continue e differenziabili rispetto a x , f e g continue rispetto a t e u..

Esempio (curva (cartesiana) di minima lunghezza)

Problemi appartenenti alla categoria del calcolo delle variazioni[23] possono essere riformulati come problemi di controllo ottimo; vediamo il passaggio di riformulazione con un semplice esempio.
Si vuole trovare una curva a minima lunghezza da un punto dato ad una linea verticale.

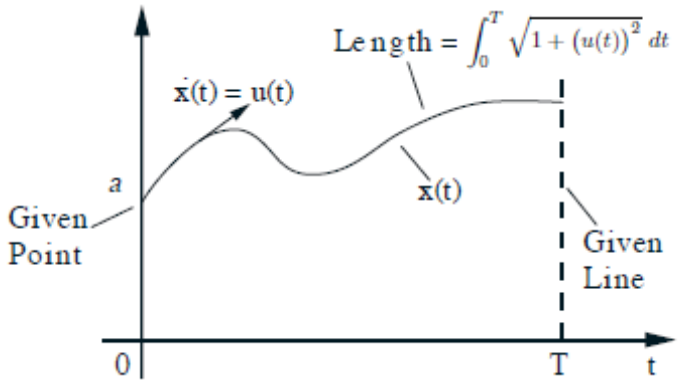

Figura 3.2: *Problema: trovare una curva a minima lunghezza da un punto dato ad una linea verticale.*

L'obiettivo è perciò quello di **minimizzare** $\int_0^T \sqrt{1+(\dot{x}(t))^2}\,dt$ [24]**dato** $x(0)=\alpha$.

La riformulazione come problema di controllo ottimo porta a introdurre il controllo *u* e l'equazione di sistema, pertanto il problema diventa:

minimizzazione di $\int_0^T \sqrt{1+(u(t))^2}\,dt$ **con** $\dot{x}(t) = u(t)$ **e** $x(0)=\alpha$.

La soluzione (intuitivamente il segmento è la curva di minima lunghezza tra due punti) viene proposta come applicazione del principio di minimo di Pontryagin[25]
(paragrafo 3.2 a pagina 30).

23 Il calcolo delle variazioni in Analisi Matematica si occupa della ricerca dei punti estremali (massimi e minimi) di funzionali cioè funzioni che hanno per dominio spazio di funzioni e per codominio $\mathbb{R}$ o $\mathbb{C}$.
24 Lunghezza di cuva cartesiana.
25 Matematico russo 1908-1988.

Descriviamo ora i passaggi per determinare la corrispondenza con l'algoritmo DP nel continuo applicando DP ad approssimazioni a tempo discreto.

Per prima cosa partizioniamo l'intervallo [0,T] nelle suddivisioni $0, \delta, 2\delta, , , , , N\delta$ con $\delta = T/N$ e poniamo $x_k = x(k\delta)$, $u_k = u(k\delta)$, $k = 0,1,\ldots,N$.

Partizioniamo anche sistema e costo $x_{k+1} = x_k + f(x_k, u_k)\delta$, $h(x_N) + \sum_0^{N-1} g(x_k, u_k)\delta$.

Scriviamo ora l'algoritmo DP per il problema riconvertito a tempo discreto,

$$\tilde{J}^*(N\delta, x) = h(x)$$
$$\tilde{J}^*(k\delta, x) = \min_{u \in U}[g(x,u)\delta + \tilde{J}^*((k+1)\delta, x + f(x,u)\delta)]$$

Assumendo $\tilde{J}^*$ differenziabile e sviluppando con Taylor al primo ordine attorno a $(k\delta, x)$:

$$\tilde{J}^*((k+1)\delta, x + f(x,u)\delta) = \tilde{J}^*(k\delta, x) + \nabla_t \tilde{J}^*(k\delta, x)\delta + \nabla_x \tilde{J}^*(k\delta, x)' f(x,u)\delta + o(\delta) ,$$

sostituendo nell'equazione DP:

$$\tilde{J}^*(k\delta, x) = \min_{u \in U}[g(x,u)\delta + \tilde{J}^*(k\delta, x) + \nabla_t \tilde{J}^*(k\delta, x)\delta + \nabla_x \tilde{J}^*(k\delta, x)' f(x,u)\delta + o(\delta)]$$

cancelliamo $\tilde{J}^*(k\delta, x)$ ambo i membri, dividiamo per δ e consideriamo il limite per $\delta \to 0$, poniamo allora $J^*(t,x)$ come costo ottimo per il problema a tempo continuo e assumiamo valido il seguente limite $\lim_{k \to \infty, \delta \to 0, k\delta = t} \tilde{J}^*(k\delta, x) = J^*(t,x), \quad \forall t, x$, otteniamo un'equazione per *J**:

Equazione di Hamilton-Jacobi-Bellman (HJB)

$$0 = \min_{u \in U}[g(x,u) + \nabla_t J^*(t,x) + \nabla_x J^*(t,x)' f(x,u)] \quad \forall t, x \quad con \quad J^*(T,x) = h(x).$$

HJB è un'equazione differenziale parziale soddisfatta per ogni coppia (t,x) dalla funzione costo $J^*(t,x)$ assumendo tale funzione differenziabile (non ci è infatti possibile stabilirlo a priori). Possiamo dimostrare formalmente che ,se l'algoritmo DP è eseguibile (le richieste computazionali non sono proibitive), allora si ha la scelta ottima ricavando il valore minimo da HJB.

La proposizione seguente stabilisce che DP è l'equivalente a tempo discreto di HJB.

Teorema (condizione sufficiente):

Supponiamo V(t,x) essere una soluzione dell'equazione HJB. V è differenziabile con continuità in t e x e tale che per ogni t e x

$$0 = \min_{u \in U}[g(x,u)+\nabla_t V(t,x)+\nabla_x V(t,x)' f(x,u)]$$

$$V(T,x)=h(x) \qquad \forall x$$

supponiamo che $\mu^*(t,x)$ ammetta minimo per ogni t e x. Poniamo $\{x^*(t)|t\in[0,T]\}$ e $u^*(t)=\mu^*(t,x^*(t)),\ t\in[0,T]$ come stato corrispondente e traiettoria di controllo,

quindi $x^*(0)=x(0),\ \dot{x^*}(t) = f(x^*(t),\mu^*(t,x^*(t))),\qquad \forall t\in[0,T]$

Allora vale $\boldsymbol{V(T,x)=J^*(t,x)}$ $\forall t,x$ e $\{u^*(t)|t\in[0,T]\}$ è ottimale.

Dimostrazione:

Ammettiamo $\{(\hat{u}(t),\hat{x}(t)|t\in[0,T])\}$ essere una traiettoria ammissibile. Per ogni $t\in[0,T]$ vale

$$0 \le g(\hat{x}(t),\hat{u}(t))+\nabla_t V(t,\hat{x}(t))+\nabla_x V(t,\hat{x}(t))' f(\hat{x}(t),\hat{u}(t)).$$

usando l'equazione di sistema $\dot{\hat{x}}(t) = f(\hat{x}(t),\hat{u}(t))$, riscriviamo il secondo membro come

$g(\hat{x}(t),\hat{u}(t))+\frac{\partial}{\partial t}[V(t,\hat{x}(t))]$ e integrando $0\le\int_0^T g(\hat{x}(t),\hat{u}(t))dt+V(T,\hat{x}(T))-V(0,\hat{x}(0))$.

Usando $V(T,x)=h(x)$ e $\hat{x}(0)=x(0)$, abbiamo $V(0,x(0))\le h(\hat{x}(T))+\int_0^T g(\hat{x}(t),\hat{u}(t))dt$,

se ora sostituiamo $u^*(t)$ e $x^*(t)$ al posto di $\hat{u}(t)$ e $\hat{x}(t)$ otteniamo uguaglianze:

$$V(0,x(0))=h(x^*(T))+\int_0^T g(x^*(t),u^*(t))\,dt \ .$$

Quindi il costo corrispondente a $\{u^*(t)|t\in[0,T]\}$ è $V(0,x(0))$ a valore minore o uguale a quello di qualsiasi altra traiettoria di controllo ammissibile $\{(\hat{u}(t)|t\in[0,T])\}$.

Segue che $\{u^*(t)|t\in[0,T]\}$ è la scelta ottimale e $V(t,x)=J^*(t,x)$ $\forall t,x$ Q.E.D

Esempio (controllo ottimo LQ – Lineare Quadratico)

L'equazione HJB è un'equazione alle derivate parziali del primo ordine che richiede ad ogni punto (x,t) la risoluzione di un problema di minimo. La sua trattazione risulta pertanto semplificata quando il problema di minimo può essere risolto in forma analitica; è il caso dei sistemi lineari a costo quadratico (LQ).
Il controllo ottimo LQ si ha nel caso in cui il sistema dinamico da controllare è di tipo lineare e le funzioni che compaiono nell'indice di comportamento sono quadratiche. Il caso LQ rappresenta uno dei problemi più significativi della teoria del controllo.

Consideriamo il sistema n-dimensionale $\dot{x}(t) = Ax(t)+Bu(t)$ ed il costo quadratico $x(T)'Q_T x(T)+\int_0^T (x(t)'Qx(t)+u(t)'Ru(t))dt$ [26], scrivendo l'equazione HJB otteniamo

$$0 = \min_{u\in\mathbb{R}^m}[x'Qx+u'Ru+\nabla_t V(t,x)+\nabla_x V(t,x)'(Ax+Bu)] \quad \text{con} \quad V(T,x)=x'Q_T x \ .$$

Vogliamo trovare una soluzione della forma $V(t,x)=x'K(t)x \qquad K(t):n\times n \ \ simmetrica$ e verificare che V(t,x) risolve HJB.

Abbiamo $\nabla_x V(t,x)=2K(t)x$ e $\nabla_t V(t,x)=x'\dot{K}(t)x$ ($\dot{K}(t)$ è la matrice che ha per elementi le derivate temporali degli elementi di K(t)). Sostituendo nell'equazione sopra

$$0 = \min_u[x'Qx+u'Ru+x'\dot{K}(t)x+2x'K(t)Ax+2x'K(t)Bu] \ .$$

il minimo in u si ottiene tramite annullamento del gradiente: $2B'K(t)x+2Ru=0$ quindi $u=-R^{-1}B'K(t)x$. Sostituiamo il valore minimo nell'equazione precedente:

$$0 = x'(\dot{K}(t)+K(t)A+A'K(t)-K(t)BR^{-1}-B'K(t)+Q)x, \quad \forall(t,x) \ .$$

Affinché $V(t,x)=x'K(t)x$ risolva HJB, K(t) deve soddisfare l'equazione seguente (**equazione di Riccati** a tempo continuo):

$\dot{K}(t)=-K(t)A-A'K(t)+K(t)BR^{-1}B'K(t)-Q$ con condizione finale $K(T)=Q_T$.

Usando la condizione sufficiente concludiamo che la funzione costo ottimo è $J^*(t,x)=x'K(t)x$

Una scelta ottima per il controllo risulta $\mu^*(t,x) = -R^{-1}B'K(t)x$.

26 A e B sono matrici note, Q_T e Q sono matrici nxn simmetriche semidefinite positive, R è matrice mxm simmetrica definita positiva.

3.2 Principio del minimo di Pontryagin

Il principio del minimo di Pontryagin rappresenta l'estensione della condizione necessaria per l'esistenza di un punto estremo (minimo o massimo) locale di una funzione di variabile reale al problema del minimo (o del massimo) di un funzionale.

In riferimento ad HJB

$$0 = \min_{u\in U}[g(x,u)+\nabla_t J^*(t,x)+\nabla_x J^*(t,x)' f(x,u)] \qquad \forall t,x \quad con \quad J^*(T,x)=h(x).$$

la funzione $J^*(t,x)$ soddisfa l'equazione sotto certe condizioni ed il teorema di verifica afferma che per un dato stato iniziale x(0) la traiettoria di controllo $\{u^*(t)|t\in[0,T]\}$ con lo stato corrispondente $\{x^*(t)|t\in[0,T]\}$ è ottima . Allora per ogni $t\in[0,T]$ vale

$$\boldsymbol{u^*(t) = \underset{u\in U}{argmin}[g(x^*(t),u)+\nabla_x J^*(t,x^*(t))' f(x^*(t),u)]} \ .$$

Il principio di minimo si basa sull'equazione precedente e la sua applicabilità dipende dal calcolo di $\nabla_x J^*(t,x^*(t))$. Per una esposizione semplificata del principio di minimo useremo il seguente lemma per il quale viene omessa la dimostrazione.

Lemma:

Data la funzione F(t,x,u) di $t\in\mathbb{R}$, $x\in\mathbb{R}^n$, $u\in\mathbb{R}^m$ differenziabile con continuità e U sottoinsieme convesso di $\mathbb{R}^m$, assumiamo che $\mu^*(t,x)$ sia differenziabile con continuità e tale che $\mu^*(t,x) = \underset{u\in U}{argmin}[F(t,x,u)] \ \forall t,x$ .Allora

$$\nabla_t\{\min_{u\in U}[F(t,x,u)]\}=\nabla_t F(t,x,\mu^*(t,x)) \ \forall t,x$$

$$\nabla_x\{\min_{u\in U}[F(t,x,u)]\}=\nabla_x F(t,x,\mu^*(t,x)) \ \forall t,x \ .$$

Differenziamo ora i membri dell'equazione HJB rispetto a x e t e annulliamo il gradiente della funzione $g(x,\mu^*(t,x))+\nabla_t J^*(t,x)+\nabla_x J^*(t,x)' f(x,\mu^*(t,x))$ usando il lemma precedente per trascurare le derivazioni di $\mu^*(t,x)$ rispetto a t e x. Per ogni (t,x) otteniamo

$$0=\nabla_x g(x,\mu^*(t,x))+\nabla^2_{xt}J^*(t,x)+\nabla^2_{xx}J^*(t,x)f(x,\mu^*(t,x))+\nabla_x f(x,\mu^*(t,x))\nabla_x J^*(t,x)$$

$$0=\nabla^2_{tt}J^*(t,x)+\nabla^2_{xt}J^*(t,x)' f(x,\mu^*(t,x))$$

Le equazioni valgono per ogni coppia (t,x) , scriviamole ora per lo stato ottimale e la traiettoria di controllo $\{x^*(t),u^*(t)\}$ con $\mu^*(t,x^*(t))=u^*(t)$, $\dot{x}^*(t)=f(x^*(t),u^*(t))$, quindi il termine $\nabla^2_{xt}J^*(t,x^*(t))+\nabla^2_{xx}J^*(t,x^*(t))f(x^*(t),u^*(t))$ nell'equazione precedente è uguale alla derivata totale rispetto a t $\frac{d}{dt}(\nabla_x J^*(t,x^*(t)))$.

Analogamente $\nabla^2_{tt} J^*(t, x^*(t)) + \nabla^2_{xt} J^*(t, x^*(t))' f(x^*(t), u^*(t))$ è uguale a

$\frac{d}{dt}(\nabla_t J^*(t, x^*(t)))$.

Denotando $p(t) = \nabla_x J^*(t, x^*(t))$ e $p_0(t) = \nabla_t J^*(t, x^*(t))$ otteniamo

$\dot{p}(t) = -\nabla_x f(x^*(t), u^*(t))\, p(t) - \nabla_x g(x^*(t), u^*(t))$ nota come **equazione aggiuntiva** e $p_0(t)$ costante per ogni t.

Usando $J^*(T, x) = h(x)$ per ogni x e la definizione di p(t) si ha $p(T) = \nabla h(x^*(t))$.

Per quanto detto u*(t) verrà riscritta come $u^*(t) = \underset{u \in U}{argmin}[g(x^*(t), u) + p(t)' f(x^*(t), u)]$

per ogni t in [0,T].

Per formulare il principio di minimo introduciamo la funzione di Hamilton o "Hamiltoniana" (mappa $(x, u, p) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ in valori reali) : $H(x, u, p) = g(x, u) + p' f(x, u)$.

Le equazioni scritte in precedenza diventano

$$\dot{x}^*(t) = \nabla_p H(x^*(t), u^*(t), p(t)) \quad \text{e} \quad \dot{p}(t) = -\nabla_x H(x^*(t), u^*(t)\, p(t)) \ .$$

Il principio di minimo può essere ora enunciato in termini di funzioni Hamiltoniane

Principio di minimo

Poniamo $\{u^*(t) | t \in [0, T]\}$ come traiettoria di controllo ottimo e $\{x^*(t) | t \in [0, T]\}$ la traiettoria di stato corrispondente: $\dot{x}^*(t) = f(x^*(t), u^*(t))$ dato $x^*(0) = x(0)$.

Assumiamo inoltre p(t) come soluzione dell'equazione aggiuntiva

$\dot{p}(t) = -\nabla_x H(x^*(t), u^*(t)\, p(t))$, con la condizione

$p(T) = \nabla h(x^*(t))$, dove h è la funzione costo terminale.

Allora per ogni t in [0,T] vale $u^*(t) = \underset{u \in U}{argmin}[H(x^*(t), u, p(t))]$ ed esiste C t.c.

$H(x^*(t), u^*(t), p(t)) = C \quad \forall t \in [0, T]$ (Hamiltoniana costante lungo la traiettoria ottima).

Nota: in riferimento ad HJB e alla definizione di p(t) vale l'ultima affermazione del principio perché $H(x^*(t), u^*(t), p(t)) = -\nabla_t J^*(t, x^*(t)) = -p_0(t)$, ma $p_0(t)$ è costante per quanto visto prima.

Il principio di minimo fornisce condizioni necessarie per l'ottimalità per cui tutte le traiettorie di controllo ottime soddisfano le equazioni sopra citate, ma, se una traiettoria di controllo verifica tali equazioni, non è detto che sia ottima.

Il principio può essere usato per ottenere una soluzione analitica in molti casi oppure come base per soluzioni di tipo numerico.

Riprendiamo l'esempio riferito al calcolo delle variazioni usando il principio di minimo per ottenere una soluzione analitica.

Esempio (curva (cartesiana) di minima lunghezza)

Vogliamo trovare la curva a minima lunghezza dal punto (0,α) alla linea $(T, y), y \in \mathbb{R}$.
Applichiamo ora le condizioni necessarie appena viste e scriviamo l'Hamiltoniana
$H(x, u, p) = \sqrt{1+u^2} + pu$ con equazioni aggiuntive $\dot{p}(t) = 0$ e $p(T) = 0$,
segue che $p(t) = 0 \quad \forall t \in [0, T]$. La minimizzazione dell'Hamiltoniana porta a
$u^*(t) = \underset{u \in \mathbb{R}}{argmin}[\sqrt{1+u^2}] = 0 \quad \forall t \in [0, T]$, quindi $\dot{x}(t) = 0$ per ogni t e x*(t) risulta costante.
Usando la condizione iniziale $x^*(0) = \alpha$ abbiamo $x^*(t) = \alpha \quad \forall t \in [0, T]$.
La soluzione intuibile a priori è pertanto la linea orizzontale passante per (0,α), ma ,dal momento che il principio di minimo è solo una condizione necessaria, non è garantito che la linea orizzontale sia la soluzione ottima. Ciò che assicura l'ottimalità della soluzione è la linearità del sistema di equazioni coinvolte e la convessità della funzione costo, sotto queste condizioni infatti il principio di minimo diventa una condizione necessaria e sufficiente per ottenere la soluzione ottima al problema[27].

Esempio (problema della brachistocrona)

L'esempio seguente è un problema di controllo ottimo a tempo continuo ed oltre a costituire un'estensione del principio di minimo ha avuto storicamente un ruolo fondamentale nello sviluppo del calcolo delle variazioni.
Il problema della brachistocrona (dal greco *brachys* = breve e *chronos* = tempo) consiste nel cercare, tra tutte le curve regolari che congiungono due punti dello spazio situati al altezze diverse, quella lungo la quale un punto materiale soggetto solo alla forza gravitazionale discende dalla quota più alta alla quota più bassa nel minor tempo possibile.
Poniamo A = (0,0) e B = (T,-b) con b>0. Il problema consiste nel trovare $\{x(t) \in [0, T]\}$ con x(0) = 0 e x(T) = b che renda minimo $\int_0^T \frac{\sqrt{1+(\dot{x}(t))^2}}{\sqrt{2\gamma x(t)}} dt$, dove γ indica l'accelerazione gravitazionale. $\{(t, -x(t)), t \in [0, T]\}$ indica la curva cercata, $\sqrt{1+(\dot{x}(t))^2}$ è la lunghezza della curva da x(t) a x (t+dt) mentre $\sqrt{2\gamma x(t)}$ è la velocità del punto materiale.

27 L'affermazione non viene dimostrata formalmente ma può essere verificata in due modalità: la prima è provare che una traiettoria di controllo ottimo esiste assieme al fatto che una sola traiettoria verifica le condizioni del principio di minimo (oppure che tutte le traiettorie che soddisfano le condizioni hanno uguale costo), la seconda modalità quando la funzione f è lineare in (x,u) e l'insieme di vincolo è convesso assieme alle funzioni di costo. In questi casi le condizioni del principio di minimo divengono necessarie e sufficienti per ottenere la soluzione ottima.

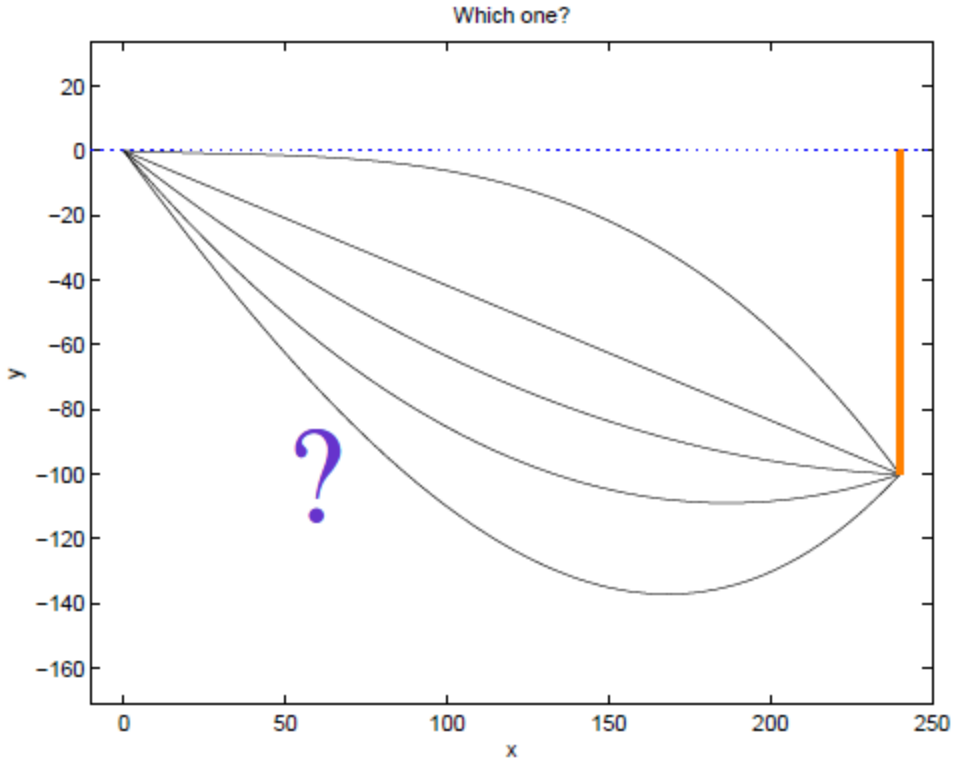

Figura 3.3 *Problema della brachistocrona: trovare la curva che nel minimo tempo permette ad un punto materiale di passare dalla quota più alta a quella più bassa sotto l'azione di gravità.*

Introduciamo il sistema $\dot{x}=u$ e poniamo $g(x,u)=\frac{\sqrt{1+(u)^2}}{\sqrt{2\gamma x}}$, l'Hamiltoniano è dato da $H(x,u,p)=g(x,u)+pu$. Ricaviamo il minimo: $p(t)=-\nabla_u g(x^*(t),u^*(t))$ [28].

Per il principio di minimo assumiamo l'Hamiltoniana costante lungo la traiettoria ottima, quindi

$$g(x^*(t),u^*(t))-\nabla_u g(x^*(t),u^*(t))u^*(t)=costante \quad \forall t\in[0,T]$$

Riscriviamo usando l'espressione di g:

$$\frac{\sqrt{1+(u^*(t))^2}}{\sqrt{2\gamma x^*(t)}}-\frac{(u^*(t))^2}{\sqrt{1+(u^*(t))^2}\sqrt{2\gamma x^*(t)}}=cost \quad o \quad \frac{1}{\sqrt{1+(u^*(t))^2}\sqrt{2\gamma x^*(t)}}=cost \quad \forall t\in[0,T].$$

Usiamo la relazione $\dot{x}^*(t)=u^*(t)$ per scrivere $x^*(t)(1+\dot{x}^*(t)^2)=C \quad \forall t\in[0,T]$ per una data costante C.

La traiettoria ottima dovrà soddisfare l'equazione $\dot{x}^*(t)=\sqrt{\frac{(C-x^*(t))}{x^*(t)}} \quad \forall t\in[0,T].$

28 Da annullamento del gradiente riscrivo p(t).

La curva ottenuta risolvendo l'equazione differenziale precedente assieme alle condizioni iniziali *x*(0) = 0 e x*(T) = b* è la cicloide e risolve il problema della brachistocrona.

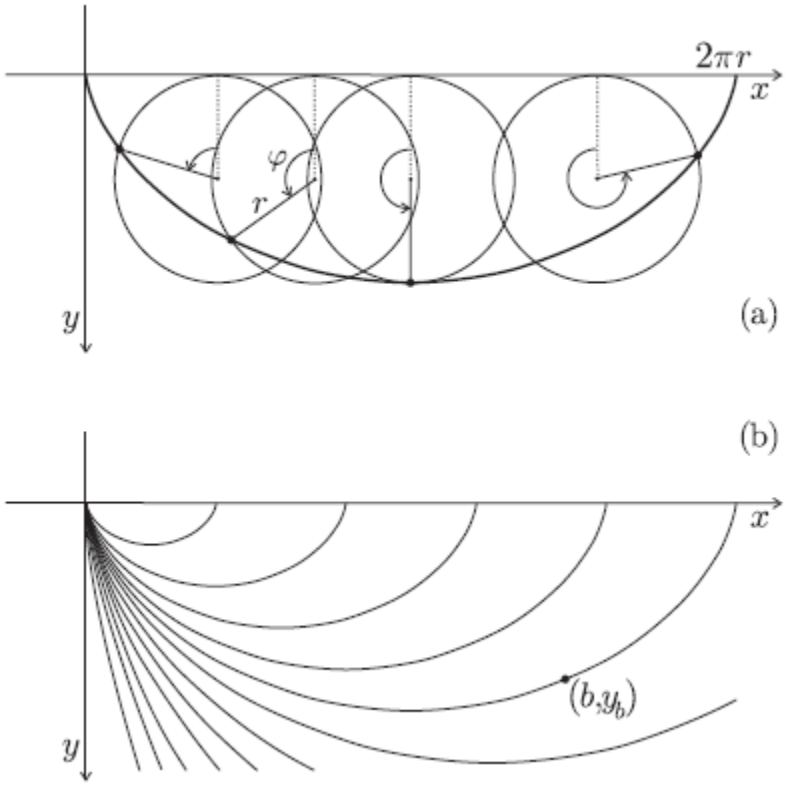

Figura 3.4: *Cicloide (a), famiglia di cicloidi con diverso r passanti per l'origine (b).*

Conclusioni

La tecnica di programmazione dinamica (DP) presenta molte similarità con i metodi risolutivi del calcolo delle variazioni.
Nei problemi di ottimizzazione sequenziale, al di là degli aspetti computazionali per i quali la tecnica DP può diventare proibitiva, essa rimane l'unico approccio valido per un problema di controllo ottimo oltre alla teoria del minimo di Pontyagin e costituisce inoltre la base per altre metodologie.
La programmazione dinamica fu composta da R. Bellman nel 1953 per risolvere in modo efficiente problemi di decisione di tipo sequenziale, in questo tipo di problemi le decisioni si attuano periodicamente e influenzano le grandezze del modello che a loro volta incidono sui processi futuri.
In questo elaborato si è cercato di approfondire il ruolo assunto dalla DP in generici problemi di ottimizzazione presentando a grandi linee gli aspetti fondanti del metodo che ne giustificano l'ampia diffusione in molti ambiti e nello specifico nella teoria del controllo.
In particolare nelle tre parti in cui il testo è suddiviso si è tentato di:

- presentare l'idea alla base della tecnica DP in maniera intuitiva per comprenderne meglio in seguito le riformulazioni equivalenti nei diversi contesti;
- sottolineare la validità del procedimento DP per problemi di controllo in ambito deterministico e non deterministico analizzando algoritmi efficienti la cui struttura trae ispirazione dal principio di ottimalità;
- esporre schematicamente alcune definizioni e teoremi fondamentali del controllo ottimo;
- Approfondire il legame tra la tecnica DP ed il calcolo delle variazioni nella soluzione di classici esercizi di ottimizzazione.

Come evidenziato dagli esempi, DP si rivela una tecnica risolutiva valida in molti contesti applicativi, ciò ne sottolinea l'importanza e il continuo riferimento in vari problemi spesso riconducibili al controllo ottimo.

Riferimenti bibliografici

[1] *Dynamic Programming and Optimal Control, volume 1,* Dimitri P. Bertsekas (2005)

[2] *Network Optimization,* Dimitri P.Bertsekas (1998)

[3] *Principles of Communications Networks and Systems,* Nevio Benvenuto, Michele Zorzi (2010)

[4] *Ottimizzazione,* Paolo Serafini (2000)

[5] *Dijkstra's algorithm revisited: the dynamic programming connexion,* articolo di Moshe Sniedovich, Dipartimento di Matematica e Statistica, Università di Melbourne, Australia (2006)

[6] *Teoria dei Sistemi e del Controllo,* dispensa di L. Biagiotti, R. Zanasi, 2010/2011

[7] *Dynamic Programming with Application,* René Caldentey (2011)

[8] *Appunti di Calcolo delle Variazioni e Controllo Ottimo,* Andrea Calogero (2012)

[9] *Appunti per il corso di Fisica Matematica,* Giancarlo Benettin (2011-2012)