Università degli Studi di Milano

FACOLTÀ DI SCIENZE E TECNOLOGIE

TESI DI LAUREA MAGISTRALE IN MATEMATICA

Algoritmi di routing ottimo per veicoli anolonomi

Relatore:

Giovanni Righini

Laureando:

Stefano Biscotti

Matricola: 962067

Anno Accademico 2020/2021

Sommario

Negli ultimi decenni la pianificazione di percorsi per veicoli è stato un campo di ricerca ampiamente studiato poiché le sue applicazioni pratiche, a partire da quelle logistiche fino ad arrivare a quelle militari, sono molteplici. Tra le numerose varianti dei problemi di cammino minimo, in questo lavoro verranno presi in esame problemi di routing adattati al caso di veicoli anolonomi come ad esempio i droni ad ala fissa. Partendo dall'analisi di problemi più semplici e già affrontati in letteratura, l'obiettivo di questa tesi è quello, oltre a fornire un'ampia panoramica sullo stato dell'arte, di individuare ed esaminare i migliori algoritmi per la risoluzione di problemi più complessi e meno dibattuti. Inizialmente verranno considerati i percorsi necessari a connettere due punti nel piano, le cui soluzioni ottime sono state individuate già alla fine degli anni '50 da Lester Dubins. In una seconda fase il problema verrà esteso dall'esigenza di connettere delle sequenze di punti, casistica che aumenta notevolmente le difficoltà nella ricerca dell'ottimo. Saranno dunque testate differenti tecniche risolutive delle quali saranno messi in luce pregi e difetti. Giunti a questo punto verrà approfondita un'ulteriore estensione; non sarà più fissata la sequenza di punti da raggiungere ma sarà noto solo l'insieme da visitare mettendoci di fronte alla variante anolonoma del celebre problema del commesso viaggiatore. Nello specifico verrà costruito un algoritmo di programmazione dinamica, di complessità esponenziale, in grado di risolvere all'ottimo il problema.

Indice

Introduzione

In questo lavoro verranno trattati problemi di routing per una particolare categoria di veicoli, quelli anolonomi. Con veicoli anolonomi (o non olonomi) intendiamo tutti quei veicoli che non hanno piena libertà di movimento nello spazio ma controllano solo una parte di tutti i possibili gradi di libertà. Consideriamo per esempio il tragitto che deve compiere un'automobile per posizionarsi in un parcheggio: chiaramente se l'automobile non è già posizionata nella direzione corretta, il percorso più breve per parcheggiare non sarà un semplice tratto dritto. Consideriamo altresì un drone che si muove nell'aria, non sarà in grado di percorrere senza fermarsi un tragitto composto da una linea spezzata ma si dovrà limitare a percorrere delle curve regolari (differenziabili). In tutti questi casi parliamo di veicoli anolonomi o veicoli di Dubins, dal nome del matematico che per primo ha trattato approfonditamente questo tipo di problemi. Gli esempi fatti non vogliono essere esaustivi della categoria che rappresentano, ma sufficienti per rendere l'idea del tipo di oggetti dei quali verranno schematizzati e modellizati i moti. In questo lavoro lo spazio di riferimento è il piano bidimensionale ma è chiaro che per esempio nel caso dei droni ci si possa estendere a lavorare in tutte e tre le dimensioni spaziali. È bene evidenziare che il modello di droni a cui facciamo riferimento è quello dei droni ad ala fissa che trovano larga applicazione ad esempio nelle mappature e nei monitoraggi di terreni. Tali veicoli, a differenza dei droni a rotore che sono invece in grado di ruotare su se stessi, sono soggetti a vincoli cinematici anolonomi che ne condizionano il tragitto.

Figura 1: A sinistra un tragitto adatto ad un drone a rotore in cui la distanza tra i punti è quella euclidea. A destra il tragitto di un drone ad ala fissa. Quest'ultimo fortemente condizionato dal raggio di curvatura minimo. [36]

Individuare le traiettorie ottimali per i veicoli anolonomi si rivela un problema per nulla banale. I modelli che ne derivano sono fortemente non lineari e spesso, nel tentativo di risolverli all'ottimo, ci si imbatte in difficoltà di convergenza o

Figura 2: Drone ad ala fissa durante un operazione di mappatura di terreni agricoli. Fonte: `https://kids.kiddle.co/Image:Agriculture_UAV.jpg`

inammissibilità. In questo lavoro viene quindi fornita un'ampia panoramica delle tecniche risolutive presenti in letteratura e vengono inoltre presentati nuovi modelli per la risoluzione di alcuni casi particolari. Va inoltre evidenziata l'importanza di saper risolvere problemi di questo genere poiché fungono da tasselli chiave per la risoluzione di modelli più complessi come ad esempio i Vehicle Routing Problems (VRP) [24] [39] [36]. I VRP, cioè i problemi di instradamento di veicoli, vengono spesso risolti tramite tecniche di Branch-and-Price. All'interno dell'algoritmo, più precisamente nel problema di pricing è necessario risolvere un Orienteering Problem (OP) [41] [26], ovvero una generalizzazione del Problema del commesso viaggiatore (TSP) in cui l'obiettivo è quello di generare un cammino che visiti un insieme di punti massimizzando i premi che la visita a ciascun punto offre e senza eccedere un prefissato budget. Lo studio delle traiettorie ottime per veicoli anolonomi trattate in questa tesi ha quindi il fine di gettare le fondamenta per la costruzione e la risoluzione di modelli più ambiziosi e complessi ai quali ci si può estendere a partire dai modelli presentati.

In letteratura il problema delle traiettorie ottimali per i veicoli anolonomi è stato trattato in maniera approfondita per la prima volta nella seconda metà degli anni '50 da Lester Dubins [9] che è stato in grado di mostrare le caratteristiche geometriche del cammino ottimo tra due configurazioni. Qualche anno più tardi risultati simili sono stati ottenuti tramite la teoria del controllo ottimo utilizzando il principio di Pontryagin e ampliati anche al caso tridimensionale [37]. È stata poi studiata da Reeds e Shepp [29] nei primi anni '90 un'estensione al problema nel caso in cui il mezzo anolonomo è in grado di muoversi, oltre che in avanti, anche all'indietro. Vale la pena sottolineare analogie e differenze dei due problemi che sono per certi aspetti molto simili ma per contro presentano alcune caratteristiche focali differenti. Innanzitutto sia i veicoli di Dubins che quelli di Reeds-Sheeps possiedono la proprietà della controllabilità. È evidente infatti, sia intuitivamente,

sia analizzando per via geometrica, che ogni punto nel piano possa essere raggiunto a partire da un punto qualunque mediante una serie di manipolazioni. Tuttavia i veicoli di Reeds-Sheeps possiedono la cosiddetta proprietà della raggiungibilità locale in tempo breve o controllabilità locale in tempo breve (in inglese small-time local controllability)[1] mentre i veicoli di Dubins no [1]. Visto il crescente interesse in questo tipo di problemi, sono sorte, a scopi differenti, numerose varianti [33]. Sono stati presi in considerazione ad esempio il caso in cui l'ambiente è caratterizzato dalla presenza di ostacoli da aggirare [20] [43] ed il caso in cui si necessiti di cammini "confortevoli" per eventuali viaggiatori sui mezzi [2]. In quest'ultima casistica è necessario introdurre vincoli sull'accelerazione e sullo strappo[2].

Il contributo originale della tesi consiste nella costruzione e risoluzione di un modello non lineare in grado di rappresentare il problema nel caso in cui si è a conoscenza del tipo di geometrie del cammino ottimo. In tal caso i modello sarà inizializzato in modo da essere risolto senza troppe difficoltà. Per costruire tale modello è stato necessario fare uso di una particolare proprietà che possiedono i percorsi ottimi nel caso in cui la distanza fra i punti è superiore al quadruplo del minimo raggio di curvatura possibile. Dopo aver dimostrato in una maniera alternativa e più intuitiva questo risultato già noto si è poi stati in grado di costruire un algoritmo di programmazione dinamica adatto alla risoluzione all'ottimo del problema di cammino hamiltoniamo minimo su un fissato set di punti.

All'interno del lavoro, dopo aver analizzato le caratteristiche e i comportamenti dei veicoli di Dubins, ci si pone l'obiettivo di individuare e calcolare i cammini ottimali di tali veicoli analizzando diverse casistiche. Viene naturale suddividere il lavoro in una parte continua e in una discreta. Inizialmente, dopo aver richiamato tutte le nozioni e i risultati noti sull'argomento si risolve il problema continuo, cioè si tenta di individuare per via algoritmica la traiettoria ottima da seguire conoscendone le caratteristiche. Successivamente, in particolare nel secondo e terzo capitolo verrà presa in analisi anche la parte discreta. Nel secondo capitolo si comincia a ragionare sulla tecnica per individuare le caratteristiche del cammino ottimo essendo a conoscenza unicamente della sequenza di punti da raggiungere. Nel terzo capitolo invece anche la sequenza di punti sarà incognita mettendoci di fronte ad una variante del ben più noto problema del commesso viaggiatore.

[1] Diciamo che un sistema è controllabile localmente in tempo breve se dato un punto p, per ogni tempo $T > 0$ esiste un intorno $\mathcal{B}(T, p)$ di p i cui punti sono tutti raggiungibili da p in un tempo non superiore a T.

[2] In cinematica lo strappo è la derivata dell'accelerazione rispetto al tempo. Solitamente vincoli di questo tipo vengono tenuti in considerazione nell'ingegneria stradale per la costruzione di strade ed autostrade ai fini del comfort dei passeggeri sui mezzi.

Capitolo 1

Cammini di Dubins

1.1 Problema di base

Prendiamo in considerazione un veicolo che si muove nel piano in un'unica direzione con la possibilità di curvare con raggio di curvatura limitato. In geometria differenziale con raggio di curvatura si intende il reciproco della curvatura, cioè il raggio della circonferenza che approssima localmente la curva. Avere un raggio di curvatura limitato implica che le curve più strette possono essere percorse solamente attorno a circonferenze con raggio uguale al raggio di curvatura minimo r_{min}. La tripla $\langle x, y, \theta \rangle$ indica la configurazione del veicolo nel piano, dove x e y ne rappresentano le coordinate mentre θ rappresenta l'angolo di orientazione rispetto all'orizzontale. Se un mezzo si muove a velocità v lungo una circonferenza di raggio r, avrà velocità angolare $\omega = \frac{v}{r}$. L'evoluzione del sistema nel tempo è pertanto:

$$\begin{cases} \dot{x} = v \cdot cos(\theta) \\ \dot{y} = v \cdot sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

dove permane il vincolo $\frac{v}{\omega} \geq r_{min}$. D'ora in poi per leggerezza di notazione chiameremo semplicemente r il raggio di curvatura minimo.

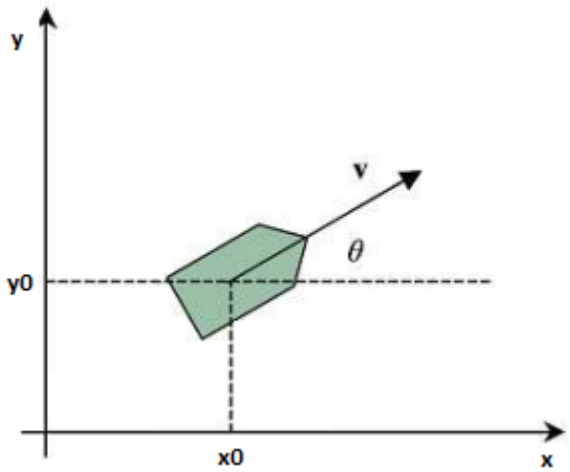


Figura 1.1: Veicolo di Dubins

Il problema è quello di ottimizzare il funzionale $J[\mathbf{X}(t)] = \int_0^{t_f} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}$ dove t_f rappresenta il tempo di arrivo (chiaramente libero) in cui viene raggiunta la posizione finale $u_f = \langle x_f, y_f, \theta_f \rangle$. È evidente che nel caso in cui la velocità sia costante il problema risulti essere un problema di cammino minimo.

1.2 Percorso più breve per due punti

1.2.1 Esistenza del percorso più breve tra due punti

Cerchiamo in questo paragrafo di riassumere i risultati raggiunti da Dubins [9] con l'obiettivo di mostrare l'esistenza e la geometria del cammino continuo e differenziabile più veloce che inizia dal punto u e termina al punto v in modo che i vettori velocità nei punti di partenza ed arrivo siano rispettivamente U e V. Si hanno inoltre i vincoli di anolonomicità cioè il veicolo è in grado si muoversi solo in avanti con la possibilità di curvare (a destra o a sinistra) con un raggio di curvatura maggiore o uguale ad un valore fissato r. Consideriamo inoltre che la velocità sia costante, in questo modo il problema, come già accennato risulta essere un problema di cammino minimo.

Consideriamo quindi un cammino nello spazio euclideo n-dimensionale E_n, che inizia dal punto u e termina al punto v e con le caratteristiche sopra elencate. Siamo interessati a trovare un cammino che soddisfi tali requisiti e che sia di lunghezza minima.

Definizione 1.2.1. *Tale cammino di lunghezza minima prende il nome* ***r-geodetica****.*

Sia X una curva nello spazio reale Euclideo $n-$dimensionale E_n (a noi basterà $n = 2$) parametrizzata rispetto alla lunghezza d'arco.

Definizione 1.2.2. *Sia la* ***curvatura media*** *di X nell'intervallo $[s_1, s_2]$ il valore* $\frac{\|X'(s_1) - X'(s_2)\|}{|s_1 - s_2|}$.

Definizione 1.2.3. *Si dice che una curva X* ***ha curvatura media sempre minore o uguale a r^{-1}*** *se X' esiste sempre e vale $\|X'(s_1) - X'(s_2)\| \leq r^{-1}|s_1 - s_2|$ per ogni s_1 e s_2 nell'intervallo di definizione di X.*

Siano u e v punti e U e V vettori nello spazio Euclideo n-dimensionale E_n, tali che $\|U\| = \|V\| = 1$ e $r > 0$. Sia $\mathrm{C} = \mathrm{C}(n, u, U, v, V, r)$ la collezione di curve X definite su un intervallo chiuso $[0, L]$, dove $\mathrm{L} = \mathrm{L(X)}$ varia con X tali che $\mathrm{X(s)} \in E_n$ per $0 \leq s \leq L$, $\|X'(s)\| \equiv 1$ e la curvatura media di X è ovunque minore o uguale a r^{-1}; $X(0) = u$, $X'(0) = U$, $X(L) = v$ e $X'(L) = V$.

Proposizione 1.2.1. *Per ogni n, u, U, v, V e r esiste una X in $C = C(n, u, U, v, V, r)$ di lunghezza minima.*

Dimostrazione. Chiaramente C è non vuoto. Per mostrarlo consideriamo le circonferenze di raggio r tangenti a U e V rispettivamente nei punti u e v, ovviamente possiamo sempre trovare le rette tangenti a tali circonferenze. Muovendoci tra gli archi di circonferenza e le rette tangenti siamo sempre in grado di individuare un cammino $\tilde{X} \in C$. Sia quindi $X_1 \in$ C e sia d_1 la sua lunghezza. Consideriamo poi d come la lunghezza dell'inf delle curve in C, dove chiaramente $d \leq d_1$. Esiste una successione $X_n \in$ C tale che le lunghezze d_n di X_n decrescono in maniera monotona a d. Poiché $\|X_n'\| \equiv 1$ segue che X_n' è una famiglia di funzioni uniformemente limitate. Poiché $\|X_n'(s_1) - X_n'(s_2)\| \leq r^{-1}|s_1 - s_2|$ in [0,d], segue che X_n' è una famiglia di funzioni uniformemente equicontinue in [0,d]. Dal teorema di Ascoli-Arzelà [13] segue l'esistenza di una sottosuccessione di X_n le cui derivate convergono uniformemente in [0,d] ad una funzione Y. Rinominiamo con X_n questa sottosuccessione. Si può mostrare facilmente che Y(0) = U e $\|Y(s_1) - Y(s_2)\| \leq r^{-1}|s_1 - s_2|$ per ogni s_1 e s_2 in [0,d]. Poiché

$$X_n(s) = X_n(0) + \int_0^s X_n'(t)dt = u + \int_0^s X_n'(t)dt$$

segue che per $0 \leq s \leq d$

$$\|X_n(s) - X_m(s)\| \leq \int_0^s \|X_n(t) - X_m(t)\|dt \leq d \cdot \sup_{t\in[0,d]} \|X_n'(t) - X_m'(t)\|$$

e quindi X_n converge uniformemente in [0,d] ad una funzione X. Poiché X_n' converge uniformemente a Y e X_n converge uniformemente a X segue che $X' = (\lim X_n)' = \lim X_n' = Y$. Quindi, visto che X_n' converge uniformemente a Y e X_n converge uniformemente a X. Completiamo la dimostrazione mostrando che $X(d) = v$ e $X'(d) = V$.

$$\begin{aligned}\|X(d) - v\| &\leq \|X(d) - X_n(d)\| + \|X_n(d) - v\| = \\ &= \|X(d) - X_n(d)\| + \|X_n(d) - X_n(d_n)\| \leq \|X(d) - X_n(d)\| + d_n - d\end{aligned}$$

che converge a 0 per n $\longrightarrow \infty$. Analogamente

$$\begin{aligned}\|X'(d) - V\| &\leq \|X'(d) - X_n'(d)\| + \|X_n'(d) - V\| = \\ &= \|X'(d) - X_n'(d)\| + \|X_n'(d) - X_n'(d_n)\| \leq \|X'(d) - X_n'(d)\| + r^{-1}|d_n - d|\end{aligned}$$

che ancora una volta converge a 0 per n $\longrightarrow \infty$.
A questo punto $X(d) = v$ e $X'(d) = V$ e quindi $X \in$ C con X di lunghezza minima. □

Ripercorriamo rapidamente i risultati ottenuti da Dubins con l'obiettivo di chiarificare i passaggi necessari per la derivazione delle conclusioni principali di questa sezione. Le dimostrazioni di tutte le Proposizioni enunciate qui nel seguito e tutti i passaggi completi per giungere alla dimostrazione del Teorema chiave di questo paragrafo si possono trovare dettagliatamente in [9].

Definizione 1.2.4. *Data una curva differenziabile Y definita tramite la lunghezza d'arco s, con $a \leq s \leq d$, diciamo che* ***è di tipo ALA (Arco, Linea, Arco)*** *se esistono b e c con $a \leq b \leq c \leq d$ tale che la restrizione di Y su $[a, b]$ è la parametrizzazione di un arco di circonferenza di raggio r di lunghezza minore o uguale a $\frac{1}{2}r\pi$, la restrizione su $[b, c]$ un segmento dritto e la restrizione su $[c, d]$ è ancora la parametrizzazione di un arco di circonferenza di raggio r di lunghezza minore o uguale a $\frac{1}{2}r\pi$.*

Proposizione 1.2.2. *Siano u, v, U, V vettori nello spazio Euclideo 2-dimensionale con $\|U\| = \|V\| = 1$ e sia r un numero reale positivo. Supponiamo che Y sia una curva di tipo ALA definita per $0 \leq s \leq d$ tale che $Y(0) = u, Y'(0) = U, Y(d) = v$ e $Y'(d) = V$. Allora Y è l'unica r-geodetica della collezione $C(2, u, U, v, V, r)$.*

Proposizione 1.2.3. *Sia X una curva planare con curvatura media sempre minore o uguale a r^{-1} definita per $0 \leq s \leq d \leq \frac{\pi r}{8}$, allora la collezione $C = (2, X(0), X'(0), X(d), X'(d), r)$ contiene una curva W di tipo ALA.*

A partire dalle Proposizioni 1.2.2 e 1.2.3 otteniamo il seguente risultato:

Proposizione 1.2.4. *Sia X una curva planare di lunghezza minore o uguale a $\frac{\pi r}{8}$. Allora X è una r-geodetica se e solo se X è di tipo ALA.*

Proposizione 1.2.5. *Sia X una curva planare definita su un intervallo chiuso e finito $[0, d]$ parametrizzata per lunghezza d'arco. Allora se X è una r-geodetica, è una curva continua e differenziabile che consiste in un numero finito di pezzi. Ognuno di questi pezzi è un segmento dritto oppure un arco di circonferenza di raggio r.*

In realtà quello che si può osservare e che ancora una volta possiamo trovare dettagliatamente in [9] è che le r-geodetiche sono dei tipi di curve molto particolari. Chiamando C degli archi di circonferenza di raggio r e S i tratti dritti è possibile mostrare che le curve di tipo SCS, CCS e CCCC non sono delle geodetiche, pertanto è possibile giungere al seguente teorema conclusivo di questo paragrafo:

Teorema 1.2.1. *Ogni r-geodetica planare è necessariamente una curva continua e differenziabile la quale è o (1) un arco di circonferenza di raggio r, seguito da un segmento rettilineo, seguito da un arco di circonferenza di raggio r, oppure (2) una sequenza di tre archi di circonferenza di raggio r, oppure (3) un sottocammino dei precedenti cammini di tipo (1) o (2).*

Osservazione: A questo punto possiamo enumerare tutti i tipi di percorsi ottimi; indichiamo quindi con L gli archi di circonferenza percorsi in senso antiorario (che corrispondono ad una curva a sinistra) e con R quelli in senso orario. I differenti percorsi sono 6 {LSL, RSR, LSR, RSL, RLR, LRL} a cui si aggiungono ovviamente i loro sottocammini.

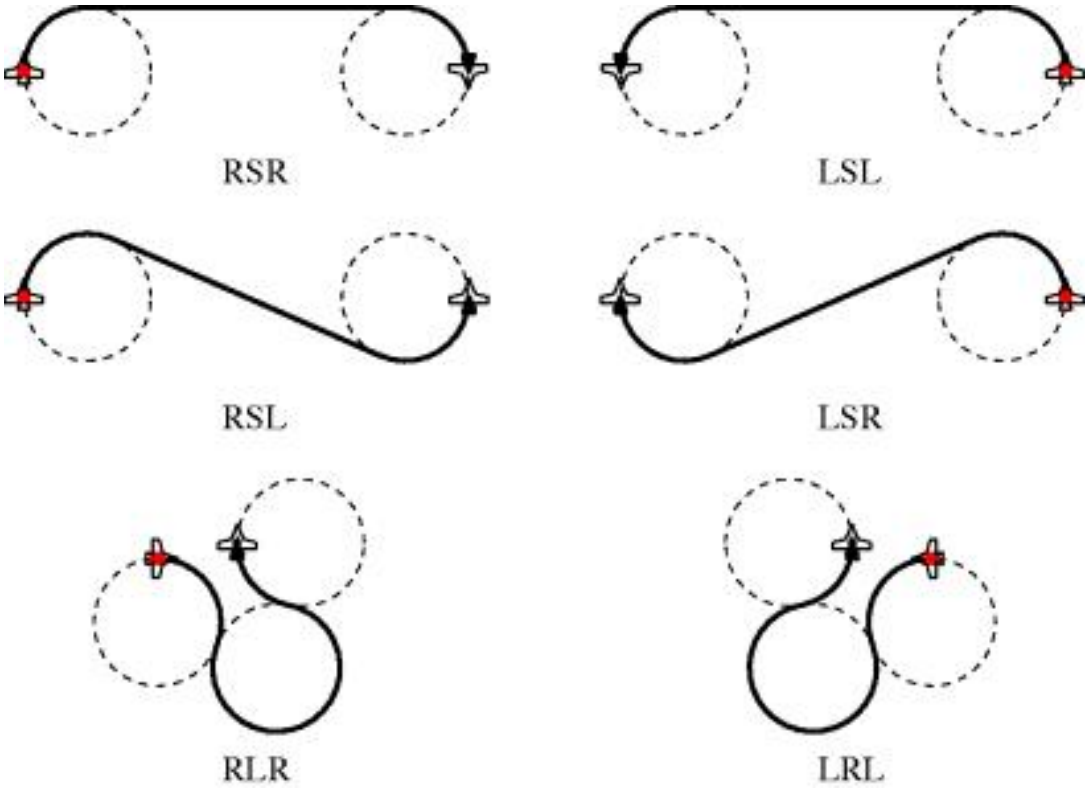


Figura 1.2: Esempio dei 6 percorsi principali

Del Teorema precedente, il quale implica che tutti i tratti curvi vengano percorsi su archi di circonferenza di raggio minimo possibile, enunciamo questo risultato intuitivo:

Teorema 1.2.2. *Siano* $(\tilde{x_0}, \tilde{y_0}, \theta_0)$ *e* $(\tilde{x_1}, \tilde{y_1}, \theta_1)$ *rispettivamente le configurazioni iniziale e finale. Sia inoltre* $\tilde{\rho} > 0$ *con* $\tilde{\rho} \neq 1$ *il raggio di curvatura minimo. Se indichiamo con* $l_{\tilde{\rho}}$ *la lunghezza del cammino di Dubins ottimo e indichiamo con* l *la lunghezza del cammino di Dubins ottimo con raggio minimo di curvatura* $\rho = 1$ *con configurazioni iniziale e finale* (x_0, y_0, θ_0) *e* (x_1, y_1, θ_1) *dove* $x_0 = \frac{\tilde{x_0}}{\tilde{\rho}}$, $x_1 = \frac{\tilde{x_1}}{\tilde{\rho}}$, $y_0 = \frac{\tilde{y_0}}{\tilde{\rho}}$ *e* $y_1 = \frac{\tilde{y_1}}{\tilde{\rho}}$, *allora* $l_{\tilde{\rho}} = \tilde{\rho} \cdot l$.

Dimostrazione. Questo risultato dipende dal fatto che i cammini ottimi sono formati sempre da tratti rettilinei e archi di circonferenza di raggio minimo. Scelte due configurazioni ed una tipologia di cammino per collegarle è chiaro che dilatare il problema (punti e raggio di curvatura) con un unico fattore di scala non ne modifica la fisionomia. Per la dimostrazione più dettagliata fare riferimento a [7]. □

Abbiamo mostrato l'esistenza e le geometrie dei cammini di Dubins, c'è da fare però un'osservazione sull'unicità, infatti quello che si osserva è che il cammino ottimo non sempre è unico. Lo si può evincere facilmente dalla seguente figura:

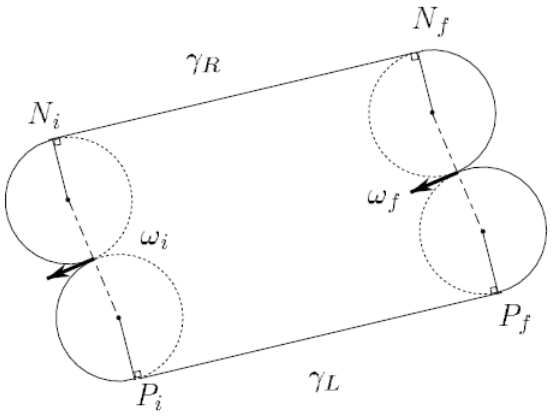


Figura 1.3: Esempio della non unicità del cammino ottimo. Nello specifico due cammini differenti con la stessa lunghezza, la migliore ottenibile per tali configurazioni che pertanto si rivelano entrambe ottime.

Questo succede ad esempio nei cammini di tipo RSR quando la somma degli archi è $2\pi r$ che possono essere rimpiazzati dai cammini isometrici LSL [5].

1.2.2 Caso di non esistenza del cammino minimo

Dopo aver mostrato l'esistenza dei cammini di Dubins, vale tuttavia la pena mostrare che sotto ipotesi più stringenti le cose cambiano.
Consideriamo ancora una volta u e v punti e U e V vettori nello spazio Euclideo n-dimensionale E_n, tali che $\|U\| = \|V\| = 1$ e $r > 0$. Sia C* = C*(n, u, U, v, V, r) la collezione di curve X definite su un intervallo chiuso $[0, L]$, dove L = L(X) varia con X tali che X(s) $\in E_n$ per $0 \leq s \leq L$, $\|X'(s)\| \equiv 1$ e $X''(s)$ esiste ovunque con $X''(s) \leq r^{-1}$ per $0 \leq s \leq L$; $X(0) = u$, $X'(0) = U$, $X(L) = v$ e $X'(L) = V$.

Proposizione 1.2.6. *Esistono u, U, v, V e r tali per cui l'inf delle lunghezze delle curve X in C* = C*$(2, u, U, v, V, r)$ non è ottenuto.*

Prima di procedere con la dimostrazione della Proposizione facciamo qualche osservazione sulle differenze tra C e C*.

Osservazione: Data X, se $X''(s)$ esiste ovunque, allora

$$\|X''(s)\| \leq r^{-1} \iff \|X'(s_1) - X'(s_2)\| \leq r^{-1}|s_1 - s_2|$$

Si intuisce quindi che C e C* corrispondono se la derivata seconda di X esiste ovunque. Quello che distingue i due insiemi è la possibilità per le funzioni in C di avere la derivata prima discontinua. Tutti i percorsi ottimi individuati in Figura 1.2 appartengono a C ma non a C* infatti nel cambio di tratto tra arco di circonferenza e rettilineo o tra due archi di circonferenza la derivata prima di X è discontinua e la derivata seconda non esiste.

Dimostrazione. Siano $u = (0, 0), U = (0, 1), v = (5, 0), V = (0, -1)$ e $r = 1$. Si mostra facilmente che esiste una curva $Y \in$ C = C$(2, u, U, v, V, r)$ composta da un arco di circonferenza di raggio $r = 1$ di lunghezza $\frac{\pi}{2}$, un segmento orizzontale di

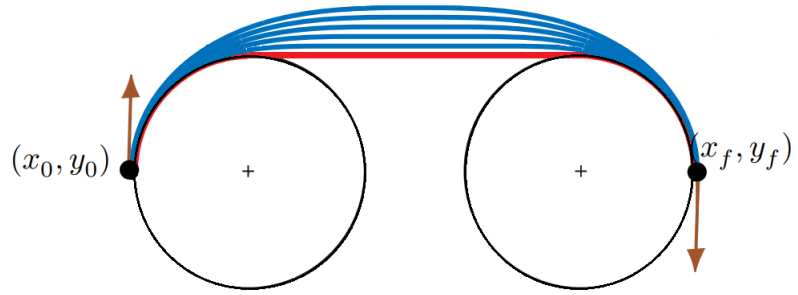


Figura 1.4: Rappresentazione grafica della non completezza di C*

lunghezza 3 ed un altro arco di circonferenza di raggio $r = 1$ di lunghezza $\frac{\pi}{2}$ che ha lunghezza totale $\pi + 3$. Dalla Proposizione 1.2.2 deriva che tale curva è l'unica curva di lunghezza minima in C.
Poiché C* $\subset$ C per completare la dimostrazione è sufficiente mostrare che per ogni $\epsilon > 0$ esiste $X \in$ C* di lunghezza minore di $\pi + 3 + 4\epsilon$. Definiamo X sull'intervallo $[0,L_\epsilon]$ specificando $L_\epsilon, X(0), X'(0)$ e la curvatura (orientata) k(s) per $0 \leq s \leq L_\epsilon$. Siano quindi $X(0) = u, X'(0) = U$ e k(s) $= 1$ per $0 \leq s \leq \frac{1}{2}\pi - \epsilon$, sia poi k(s) lineare per $\frac{1}{2}\pi - \epsilon \leq s \leq \frac{1}{2}\pi + \epsilon$ e sia k$(\frac{1}{2}\pi + \epsilon) = 0$. Sia ora d la prima coordinata del punto $X(\frac{1}{2}\pi + \epsilon)$, poniamo k(s) $= 0$ per $\frac{1}{2}\pi + \epsilon \leq s \leq 5 + \frac{1}{2}\pi + \epsilon - 2d$. A questo punto sia k(s) lineare per $5 + \frac{1}{2}\pi + \epsilon - 2d \leq s \leq 5 + \frac{1}{2}\pi + 3\epsilon - 2d$ e infine k(s) $= 1$ per per $5 + \frac{1}{2}\pi + 3\epsilon - 2d \leq s \leq 5 + \pi + 2\epsilon - 2d = L_\epsilon$.
X è unicamente definito ed è inoltre facile mostrare che $X \in$ C* e che $L_\epsilon < \pi + 3 + 4\epsilon$. Pur omettendo questi calcoli è chiaro che X è un'approssimazione due volte differenziabile di Y e che le due curve differiscono per una quantità $f(\epsilon) \to 0$ per $\epsilon \to 0$. □

Osservazione: Si potrebbe congetturare che data una qualunque r-geodetica Y in C$(2, u, U, v, V, r)$ e $\epsilon > 0$, allora esiste X in C*$(2, u, U, v, V, r)$ la cui lunghezza differisce da quella di Y di una quantità $f(\epsilon) \to 0$ per $\epsilon \to 0$. Questa proprietà non è tuttavia garantita, ne diamo pertanto un controesempio. È sufficiente considerare Y (continua e differenziabile) composta da un arco di circonferenza di raggio $r = 1$ e lunghezza $\frac{1}{2}\pi$ orientato in senso antiorario seguito da un arco di circonferenza di raggio $r = 1$ e lunghezza $\frac{1}{2}\pi$ orientato in senso orario. Si lascia al lettore la dimostrazione della validità di tale controesempio.

1.2.3 La Distanza di Dubins

Prima di capire come calcolare il percorso più breve tra due punti vale la pena spendere qualche parola introduttiva. Con Distanza di Dubins intendiamo la lunghezza necessaria a collegare due configurazioni tramite un cammino di Dubins. In realtà, come si potrebbe aver intuito, non si tratta di una vera e propria distanza poiché non possiede la proprietà della simmetria. Di conseguenza, nonostante nel seguito con un abuso di notazione parleremo di distanza, non possiamo considerarla a tutti gli effetti una metrica. Consideriamo ora il caso in cui l'angolo finale sia variabile. Come prima cosa si può notare che un punto messo in posizione ravvicinata (in senso euclideo) rispetto al precedente potrebbe avere una distanza nel senso di

Dubins maggiore rispetto ad un punto più lontano (in senso euclideo).
Esempio 1: consideriamo $u = (0,0)$ con $U = \frac{\pi}{2}$ come punto di partenza e supponiamo di voler raggiungere i punti $v = (0,1)$ e $v' = (0.5,0)$. È banale constatare che il punto v è più distante da u rispetto a v' per quel che riguarda la metrica euclidea. Ragionando però tramite la distanza nel senso di Dubins, il cammino più breve risulta essere quello tra u e v.

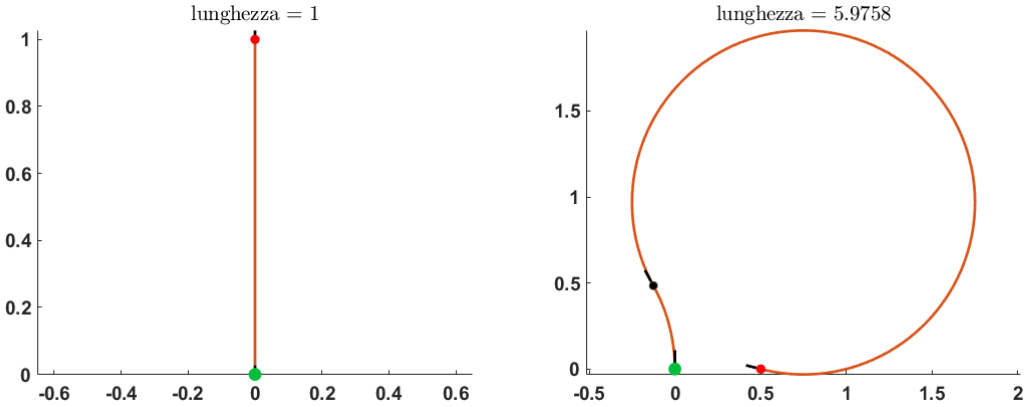


Figura 1.5: Rappresentazione dell'**esempio 1:** a sinistra il cammino ottimo tra u e v mente a destra quello tra u e v' con le rispettive lunghezze

Un'altra osservazione che si può fare sulla distanza di Dubins è il fatto che non possieda la proprietà della continuità come mostrato nell'esempio seguente.

Esempio 2: Consideriamo u e U come nell'esempio precedente mentre siano $\tilde{v} = (1,1)$ e $\tilde{v'} = (1,1\text{-}\epsilon)$ con $\epsilon > 0$ piccolo. Per raggiungere $\tilde{v}$ è sufficiente percorrere un arco di circonferenza verso destra per una lunghezza totale del cammino di $\frac{\pi}{2}$. Per raggiungere $\tilde{v'}$ invece è necessario modificare completamente la traiettoria, serve infatti compiere inizialmente un arco di circonferenza verso sinistra e solo in un secondo momento un arco di circonferenza a destra. Questo processo porta ad allungare notevolmente il percorso come si può facilmente vedere nella Figura 1.6. Nella elaborazione con il calcolatore è stato utilizzato $\epsilon = 0.00001$.

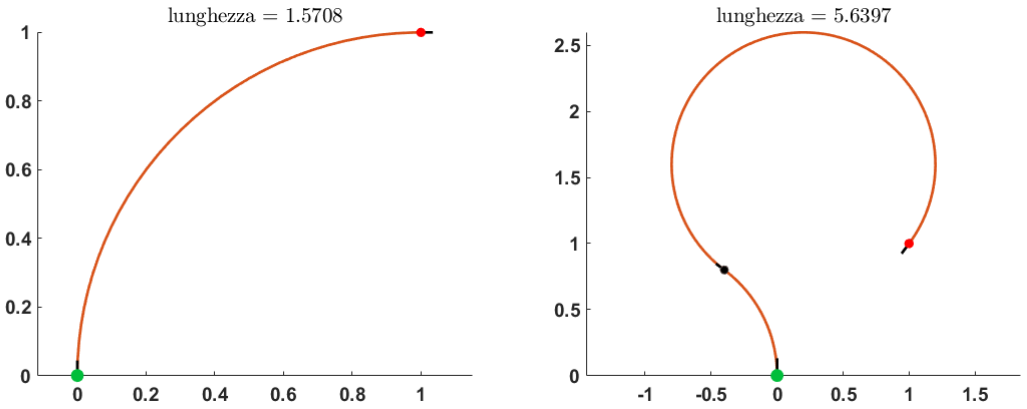


Figura 1.6: Rappresentazione dell'**esempio 2:** a sinistra il cammino ottimo tra u e $\tilde{v}$ mente a destra quello tra u e $\tilde{v'}$ con le rispettive lunghezze.

Riferendoci un'altra volta al problema di ottimizzazione in cui l'angolo di arrivo è variabile, facciamo ancora qualche considerazione. Innanzitutto, e questa osservazione vale anche nel caso in cui la direzione di arrivo non è variabile, possiamo sempre ricondurci, mediante rototraslazioni, al problema con raggio unitario in cui u = (0,0) e U = (1,0). In questo modo dovrebbe risultare chiara la simmetria generata dall'asse x; se consideriamo il punto v = (x,y) e il cammino ottimo per raggiungerlo γ, è chiaro che il cammino ottimo per giungere a $\bar{v}$ = (-x,y) sarà γ' simmetrico a γ rispetto all'asse delle ascisse. Sarebbe interessante cercare di intuire a priori il tipo di cammino ottimale da seguire per raggiungere un qualsiasi punto del piano. Mostriamo a titolo di esempio l'idea del processo da seguire per ottenere questo tipo di informazioni (per ulteriori dettagli fare riferimento a [6]). Ricordiamo che ci si può ridurre, per motivi di simmetria, a studiare solamente il semipiano superiore. Consideriamo quindi il dominio dei cammini del tipo L_uS_s (quelli simmetrici sono del tipo R_uS_s) sapendo che $u \in [0, 2\pi[$ e $s \geq 0$. Dopo aver percorso, a partire dall'origine, il primo tratto L di lunghezza u si giunge al punto $M_L = (sin(u), 1 - cos(u))$ e successivamente dopo il tratto rettilineo S di lunghezza s si giunge a $M_{LS} = (sin(u) + s \cdot cos(u), 1 - cos(u) + s \cdot sin(u))$. Al variare dei parametri u, s possono essere raggiunti tutti i punti del piano all'infuori della circonferenza unitaria centrata in (0,1).
In maniera del tutto analoga possiamo ricavare il dominio di tipo RL e la sua controparte simmetrica LR. Nelle figure seguenti sono rappresentati graficamente i domini, ricordando che nel caso R_uL_v vale, al fine dell'ottimalità, che $u \in [0, v[$ e $v \in]\pi, 2\pi[$.
Ora, poiché le parti del piano sono coperte da diverse tipologie di cammino, è necessario capire quale sia quella ottimale. Tralasciando i calcoli necessari per giungere alle conclusioni [6] riportiamo l'immagine della partizione del piano con i cammini ottimi per raggiungere ogni punto nello spazio senza una direzione fissata di arrivo.

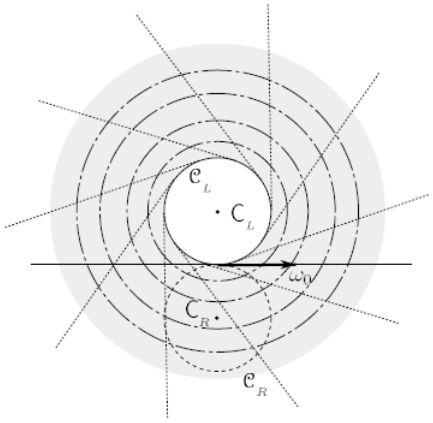


Figura 1.7: Dominio del tipo LS

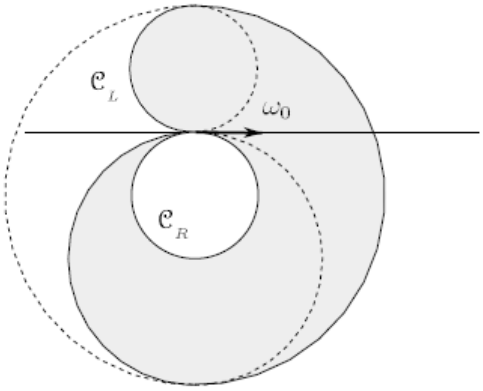


Figura 1.8: Dominio del tipo RL

C_L = (0,1) e C_R = (0,-1) sono i centri delle circonferenze di raggio unitario. Si deducono facilmente i domini dei casi simmetrici.

I punti situati sugli archi di circonferenza con x < 0 vengono raggiunti tramite un unico tratto del tipo L o R, pertanto si può considerare come forma degenere sia dei cammini RL e LR sia di quelli LS e RS. Casistica simile per i punti sull'asse

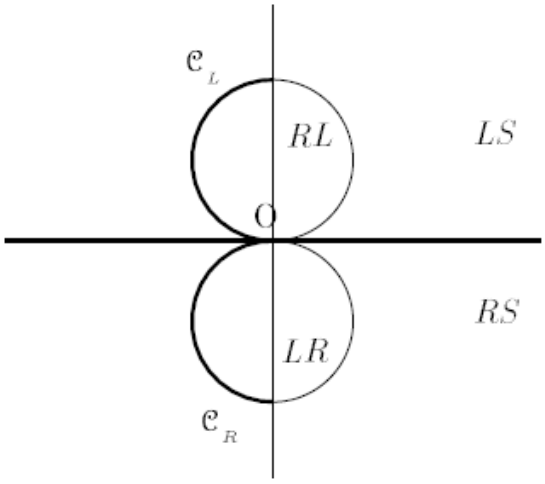


Figura 1.9: In grassetto vengono evidenziate le zone che possono essere raggiunte da due differenti cammini di uguale lunghezza

delle ascisse con x $\geq$ 0 che vengono raggiunti da un unico tratto rettilineo, caso degenere sia di LS che di RS.
A partire dalle informazioni ricavate, possiamo darci un'idea migliore di come è fatta la distanza di Dubins quando la direzione di arrivo è variabile. Si mostra in seguito una rappresentazione tramite curve di livello [6]. Dall'immagine, come in realtà si potrebbe già aver intuito, è evidente il fatto che allontanandosi dal punto di partenza, la Distanza di Dubins tende ad assomigliare a quella euclidea, cosa che invece non succede, nei pressi del punto di partenza.

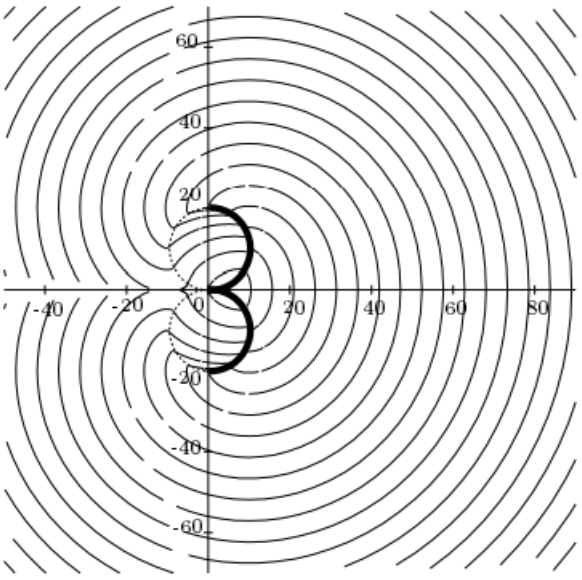


Figura 1.10: Rappresentazione grafica della Distanza di Dubins con u = (0,0) e U = (1,0). In grassetto viene evidenziata la zona di discontinuità della distanza.

1.2.4 Calcolo del cammino ottimo tra due punti

Avendo analizzato tutte le geometrie possibili dei cammini tra due punti e le caratteristiche della distanza con la direzione di arrivo libera, è interessante capire quale sia nella pratica il tipo di cammino ottimo tra due punti con direzioni di partenza ed arrivo fissate. Un primo approccio potrebbe essere quello di calcolare (quando possibile) tutti e sei i cammini per poi decretare il migliore confrontandone le lunghezze. Tuttavia è abbastanza evidente il fatto che una volta definiti l'angolo di partenza e di arrivo, la tipologia di cammino ottimo è univocamente definita. In generale, una prima osservazione che si può fare è che quando due punti sono ad una distanza euclidea maggiore di $4r$ non è possibile collegarli con tratti del tipo RLR e LRL ma è necessario almeno un segmento rettilineo all'interno del cammino. Nel caso in cui i due punti si trovino a distanze euclidee minori di $4r$ non è detto invece che si possano congiungere con tratti del tipo RSR, LSL, RSL e LSR mentre saranno certamente congiungibili da tre archi di circonferenza.
Prima di intraprendere una serie di considerazioni simili a quelle riportate nel paragrafo precedente che ci permetteranno di avere una panoramica sulle varie partizioni del piano, lasciamo un'indicazione su come costruire concretamente i vari tipi di cammini di Dubins. Senza perderci in ulteriori tecnicismi demandiamo a [11], che fornisce una guida passo passo per la costruzione e il calcolo dei cammini fornendo indicazioni per ricavare tutti i punti e i tratti necessari. A questo punto possiamo cominciare a ricavare le informazioni sui vari tipi di domini al fine di costruire una partizione in grado di darci un'indicazione a priori sul cammino ottimo tra due punti.

Ricordiamo innanzitutto la possibilità di ridurci a $r = 1$ e porre $u = (0, 0)$ con $U = (1, 0)$. Nel cercare il cammino ottimo che ci porti a $v = (x, y)$ con $V = \theta$ si può innanzitutto individuare nella retta che passa per l'origine con pendenza $\frac{\theta}{2}$ un asse di simmetria che ci permette di limitarci a studiare metà del piano [5]. Prima di addentrarci nello studio dei singoli domini è necessario mostrare una figura esemplificativa che sarà necessaria per la comprensione di quelle successive:

- E è il punto di coordinate $(sin(\theta), 1 - cos(\theta))$
- G è il punto di coordinate $(sin(\theta), -1 - cos(\theta))$
- F (risp. H) è il punto simmetrico ad E (risp. G) rispetto all'origine O
- J (risp. K) è il punto simmetrico ad H (risp. G) rispetto al punto E
- D_0 (risp. D_1) è la semiretta uscente da E nella direzione delle coordinate x positive, con orientazione 0 (risp. θ)
- D_2 (risp. D_3) è la semiretta uscente da F parallela a D_0 (risp. D_1)
- C_P è la circonferenza centrata nel generico punto P con raggio 2

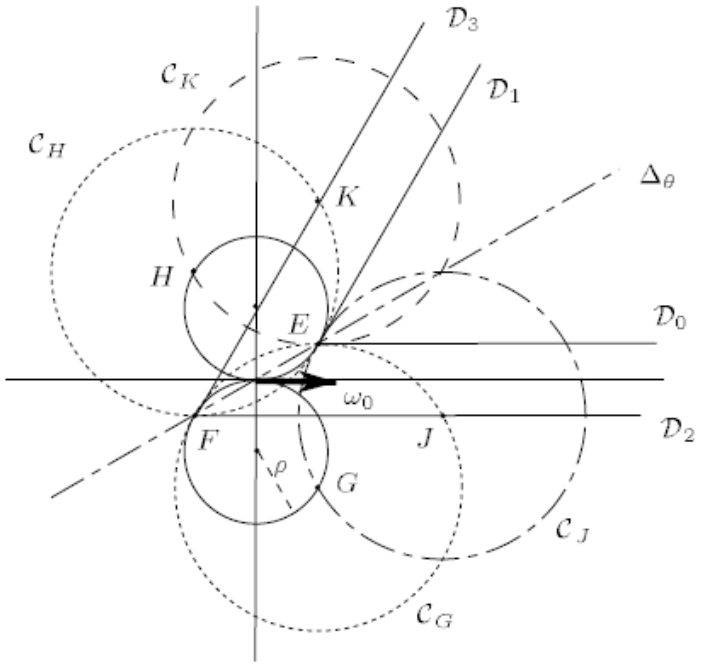


Figura 1.11: Riferimenti a punti, rette e circonferenze necessari al calcolo dei domini

Domini

Cominciamo considerando un cammino del tipo $L_v S_s L_u$ dove chiaramente $u+v = \theta$. Manteniamo (v, s) come parametri tenendo presente le condizioni di ottimalità $v, u \geq 0$ e $v + u = 2\pi$.

Seguendo il primo arco di lunghezza v giungiamo alle coordinate: $\begin{cases} sin(v) \\ 1 - cos(v) \end{cases}$

Successivamente, dopo il tratto rettilineo giungiamo in $\begin{cases} sin(v) + s \cdot cos(v) \\ 1 - cos(v) + s \cdot sin(v) \end{cases}$

Infine dopo l'ultimo tratto di lunghezza $u = \theta - v$ si giunge in

$$\begin{cases} sin(v) + s \cdot cos(v) + sin(\theta) - sin(v) \\ 1 - cos(v) + s \cdot sin(v) + cos(v) - cos(\theta) \end{cases}$$

Semplificando, il punto di arrivo di un cammino del tipo $L_v S_s L_{\theta - v}$ al variare di $v \in [0, \theta]$ e $s \in \mathbb{R}^+$ è il punto $M_{LSL} = (s \cdot cos(v) + sin(\theta), 1 + s \cdot sin(v) - cos(\theta))$. Abbiamo così ricavato il dominio dei cammini LSL che si può osservare graficamente nella figura successiva. In maniera del tutto analoga, partendo dall'origine e aggiungendo un tratto per volta, possiamo ricavare anche i domini delle altre tipologie di cammino. Di seguito, a titolo di esempio, oltre alla rappresentazione del dominio LSL è mostrata anche quella del dominio RSR. Per ulteriori approfondimenti sul calcolo di tutti i domini si veda [5]. Fissato un angolo θ possiamo ora suddividere il piano nelle varie sezioni e capire da quali tipologie di cammino sono raggiunte. Per esempio, se fissiamo $\theta = \frac{\pi}{4}$ possiamo dividere il piano in 13 sezioni ognuna raggiunta solamente da alcuni particolari cammini.

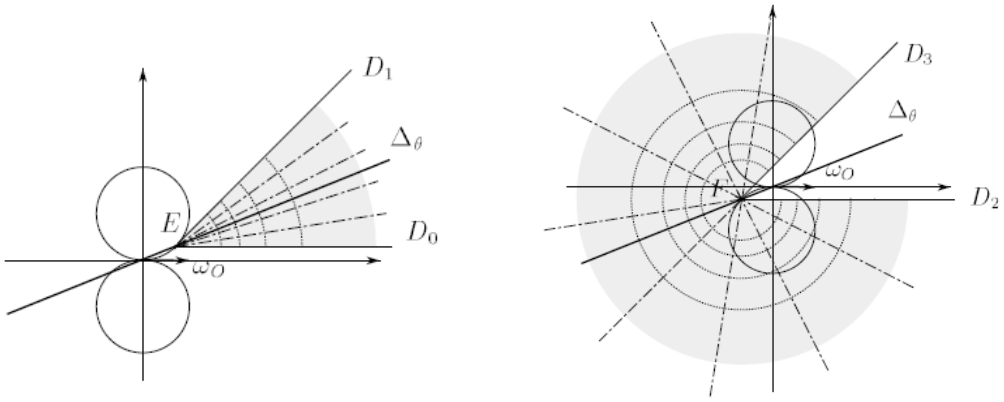


Figura 1.12: A sinistra dominio del tipo LSL mentre a destra del tipo RSR

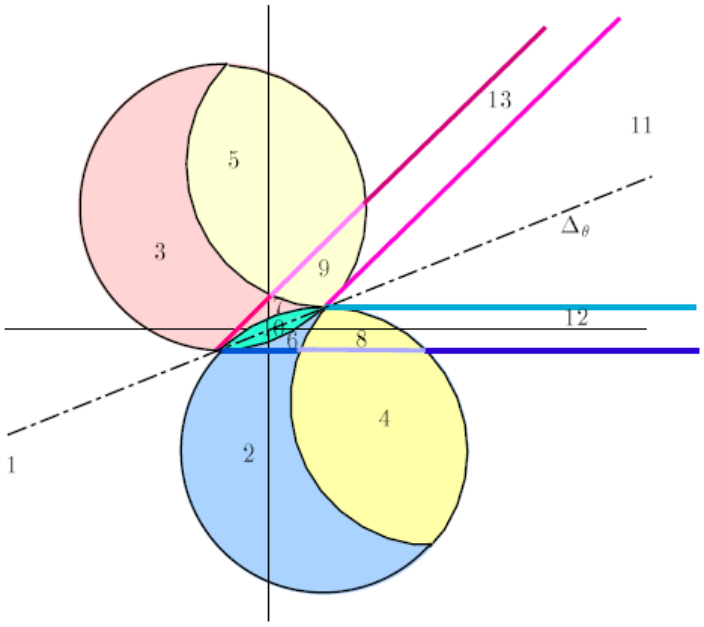


Figura 1.13: Intersezione dei domini per $\theta = \frac{\pi}{4}$

RSR: 1 - 2 - 3 - 4 - 5
LSL: 11
RLR: 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10
LRL: 4 - 5 - 8 - 9 - 10
RSL: 1 - 3 - 5 - 7 - 9 - 11 - 12 - 13
LSR: 1 - 2 - 4 - 6 - 8 - 11 - 12 - 13

L'ultimo passaggio da fare è quello di individuare per ogni settore e punto nel piano qual è il tipo di cammino ottimo, quindi in ogni settore si verifica la lunghezza dei vari cammini e si stabilisce il migliore. Chiaramente, poiché al variare di θ ogni dominio è differente, questa operazione andrebbe compiuta nelle varie casistiche.

Nel lavoro di Bui et al. [5] si possono trovare nel dettaglio i vari confronti tra le tipologie di cammino e infine alcune partizioni per angoli noti. Qui in seguito la figura esemplificativa della partizione nel caso in cui $\theta = \frac{\pi}{5}$.

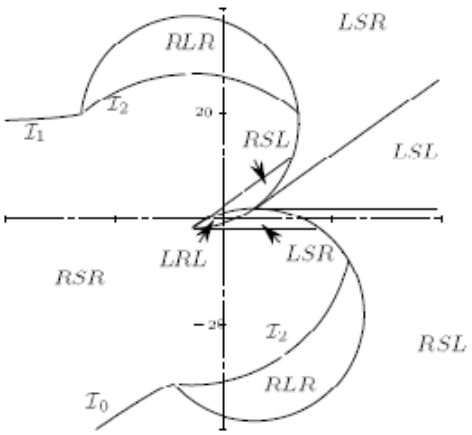


Figura 1.14: Esempio di partizione di $P_{\pi/5}$

Dopo quest'analisi, dovrebbe risultare chiaro il fatto che dati due punti con le rispettive direzioni, il cammino di Dubins ottimo è definito e calcolabile in tempo costante. In realtà queste modalità di ottenimento del cammino ottimo appena descritte non verranno utilizzate direttamente. Visto il crescente interesse nello studio dei cammini di Dubins, è stato introdotto nella release R2019b di Matlab, all'interno del Navigation Toolbox, l'oggetto `dubinsConnection` con la relativa funzione `connect`. Questa funzione, confrontando tutte le lunghezze dei cammini tra due configurazioni di partenza ed arrivo, è in grado di generare il cammino di Dubins ottimo. È poi possibile ottenere informazioni sul suo tipo e sulle varie lunghezze dei segmenti. Sono inoltre disponibili opzioni per modificare il raggio di curvatura massimo oppure per escludere determinati tipi di curve dalla ricerca dell'ottimo.

Capitolo 2

Cammini minimi per sequenze di punti

2.1 Percorso più breve per tre o più punti

Una volta mostrata l'esistenza del cammino ottimo ed aver capito la strategia per ottenerlo, ci poniamo l'obiettivo di ampliare il problema nel caso in cui i punti da raggiungere (escluso il punto di partenza) siano in numero N $>$ 1. In questo capitolo prendiamo in considerazione il caso in cui i punti da sorvolare sono forniti in una sequenza prefissata. Il problema di trovare il cammino di Dubins passante per una sequenza di punti è l'estensione naturale del problema di Dubins per due punti che prende il nome di interpolazione di Markov-Dubins[1] [18].
Anche se all'apparenza questo problema potrebbe sembrare banale, è evidente ad un'osservazione più attenta che non è sufficiente individuare il percorso per più punti concatenando percorsi ottimi tra due punti consecutivi. Questo è dovuto al fatto che all'aggiunta di ogni punto, potrebbe variare, seppur leggermente, anche la direzione di passaggio nei punti precedenti. Consideriamo quindi la curva X:$[0, t_N] \to \mathbb{R}^2$ tale che passi, in maniera ordinata, per gli N + 1 punti $u_0, u_1, ..., u_N$ ai tempi $0 < t_1 < ... < t_N$, dove il vettore velocità è fissato nei punti u_0 e u_N. Ricordiamo che i parametri $t_1, t_2, ..., t_N$ sono incogniti, pertanto possiamo riformulare il problema nella notazione della teoria del controllo ottimo che ci permette chiarezza e compattezza:

$$(\mathrm{P}) \begin{cases} min & t_n \\ s.t. & X(t_0) = u_0, X(t_1) = u_1, ..., X(t_N) = u_N \\ & X'(t_0) = U_0, X'(t_N) = U_N \\ & \|X''(t)\| \leq r^{-1}, \|X'(t)\| = 1 \ \ per\ t \in [0, t_N] \end{cases}$$

oppure, con una notazione ancor più maneggevole:

[1] Andrey Andreyevich Markov è il nome del matematico russo che per primo studiò alcune istanze del problema nel 1889 [18].

$$(\text{Pc}) \begin{cases} min & t_n \\ s.t. & \dot{x}(t) = cos\theta(t), x(0) = x_0, x(t_1) = x_1, ..., x(t_N) = x_N \\ & \dot{y}(t) = sin\theta(t), y(0) = y_0, y(t_1) = y_1, ..., y(t_N) = y_N \\ & \dot{\theta}(t) = u(t), \theta(0) = \theta_0, \theta(N) = \theta_N \\ & |u(t)| \leq r^{-1}, \ \ per\ t \in [0, t_N] \end{cases}$$

2.1.1 Proprietà discrete

Analizzando il problema siamo in grado di dedurre alcune importanti caratteristiche. Innanzitutto, per quanto a questo punto possa già sembrare intuitivo, si può generalizzare il Teorema 1.2.1.

Teorema 2.1.1. *Ogni soluzione del problema (P), cioè una curva C^1 e C^2 a tratti tale che costituisce il cammino minimo di curvatura massima limitata che percorre in maniera ordinata una sequenza di punti $u_0, ..., u_N$, è di tipo CCC o CLC (o un loro sottoinsieme) tra due punti consecutivi. Se inoltre tra due punti il cammino ottimo è del tipo CCC, il secondo arco di circonferenza ha lunghezza maggiore di πr.*

Dimostrazione. Si veda [18]. □

Proposizione 2.1.1. *Se due punti consecutivi u_i e u_{i+1} si trovano a distanza maggiore o uguale a $4r$ allora il cammino di Dubins più corto tra le configurazioni (u_i, θ_i) e (u_{i+1}, θ_{i+1}) è del tipo CLC o un suo sottoinsieme.*

Dimostrazione. Si deduce dalle osservazioni sui domini mostrate nel paragrafo 4.3 di [5]. □

Osservazione: Quando la distanza tra u_i e u_{i+1} è esattamente uguale a $4r$ può succedere che il cammino minimo sia della forma degenere CC, cioè CLC con il tratto L di lunghezza nulla.

2.1.2 Proprietà continue

Proposizione 2.1.2. *Sia dato il cammino ottimo che risolve il problema (P), se due tratti consecutivi (da $i-1$ a i e da i a $i+1$) sono del tipo CLC allora il punto i divide l'arco C in due sottoarchi di uguale lunghezza. Inoltre se nel tratto da $i-1$ a i indichiamo con θ_C l'orientazione della velocità all'inizio dell'ultima curva e θ_i l'orientazione della velocità al punto i, $|\theta_i - \theta_C| < \pi$.*

Dimostrazione. Si veda Proposizione 2 [18]. Per una dimostrazione alternativa della prima parte senza fare uso della teoria del controllo ottimo si faccia invece riferimento al Lemma 4.1 (iii) di [12]. □

Dimostrazione geometrica - intuitiva

Cerchiamo di mostrare in maniera geometrica e sicuramente più intuitiva il fatto che dati due tratti consecutivi del tipo CSC, l'arco di circonferenza finale del primo tratto e quello iniziale del secondo hanno lunghezza identica. Consideriamo dunque tre punti consecutivi collegati da due tratti di tipo CSC. Chiamiamo dunque C_1 e

C_2 i centri delle circonferenze di raggio unitario (fissate) su cui poggiano il primo e l'ultimo arco di circonferenza. Sia invece C il centro della circonferenza "centrale" da fissare su cui poggiano il punto di passaggio P e gli archi di circonferenza intermedi. Chiaramente C è vincolato su una circonferenza di raggio unitario centrata in P. Possiamo immaginare questo problema come un problema di equilibrio, dove le circonferenze sono delle pulegge, la prima e la terza fissate e quella centrale vincolata ad avere il centro a distanza 1 da P. Il cammino è rappresentato da un elastico che tenderà all'equilibrio stabile che è il cammino ottimo. Consideriamo ora il cammino che passa per i tre punti consecutivi, le curve del cammino possono essere percorse in differenti direzioni (R o L) ma la dimostrazione non cambia. Nella dimostrazione si articolano due parti successive: inizialmente si individua la posizione di C per minimizzare i tratti rettilinei mentre in una seconda fase si procede alla minimizzazione degli archi curvi. Indichiamo con A_1 e A_2 i punti di tangenza dei cammini su C e con B_1 e B_2 i punti di tangenza sulle circonferenze C_1 e C_2. Volendo minimizzare la somma dei segmenti A_1B_1 e A_2B_2, il gradiente dell'obiettivo è la somma di due vettori unitari orientati lungo i rispettivi segmenti A_1B_1 e A_2B_2 con verso da A_1 a B_1 e da A_2 verso B_2. Per la regola del parallelogramma, la loro somma è sulla bisettrice delle due rette che si ottengono prolungando i segmenti A_1B_1 e A_2B_2. L'ottimizzazione è vincolata al fatto che C passi per il punto P e che $\overline{CP} = 1$. Il gradiente di tale vincolo giace sulla retta CP. Per le condizioni necessarie del primo ordine, si ha l'ottimo solamente se gradiente dell'obiettivo e gradiente del vincolo sono allineati. All'ottimo pertanto vale la proprietà prevista. Occupiamoci ora della minimizzazione degli archi percorsi; minimizzare gli archi è equivalente a massimizzare la lunghezza del segmento CH dove H è l'intersezione del prolungamento dei segmenti A_1B_1 e A_2B_2. Il gradiente di tale obiettivo giace proprio sulla retta CH. Massimizzando tale segmento le tangenti tendono ad allinearsi riducendo la lunghezza dell'arco centrale. Poiché come nel caso precedente l'ottimizzazione è vincolata al fatto che C passi per il punto P e che $\overline{CP} = 1$, ancora una volta per le condizioni necessarie del primo ordine otteniamo che P nell'ottimo divide il suo arco in due sottoarchi di ugual lunghezza. Per concludere, mostriamo che se la tesi vale per le sequenze di tre circonferenze consecutive con le circonferenze esterne fissate e la centrale da fissare, allora vale necessariamente per la sequenza ottima. Se infatti esistesse lungo la sequenza una circonferenza che non soddisfa la condizione, allora la soluzione potrebbe essere migliorata muovendo adeguatamente quella sola circonferenza e quindi non sarebbe ottima.

Le proprietà individuate dalle Proposizioni 2.1.1 e 2.1.2 permettono, in un caso standard in cui i punti sono "abbastanza" distanziati l'uno dall'altro, di asserire che il cammino di Markov-Dubins ottimo è della forma CLCLC...CLCLC dove i punti di passaggio dividono gli archi curvi in due tratti di lunghezza uguale.

Osservazione: Se le direzioni iniziale e finale sono libere, il cammino comincia e termina con un tratto di tipo L. Contrariamente, se le direzioni sono fissate il cammino inizierà e terminerà con tratti di tipo C che potrebbero tuttavia avere in alcuni casi lunghezza nulla.

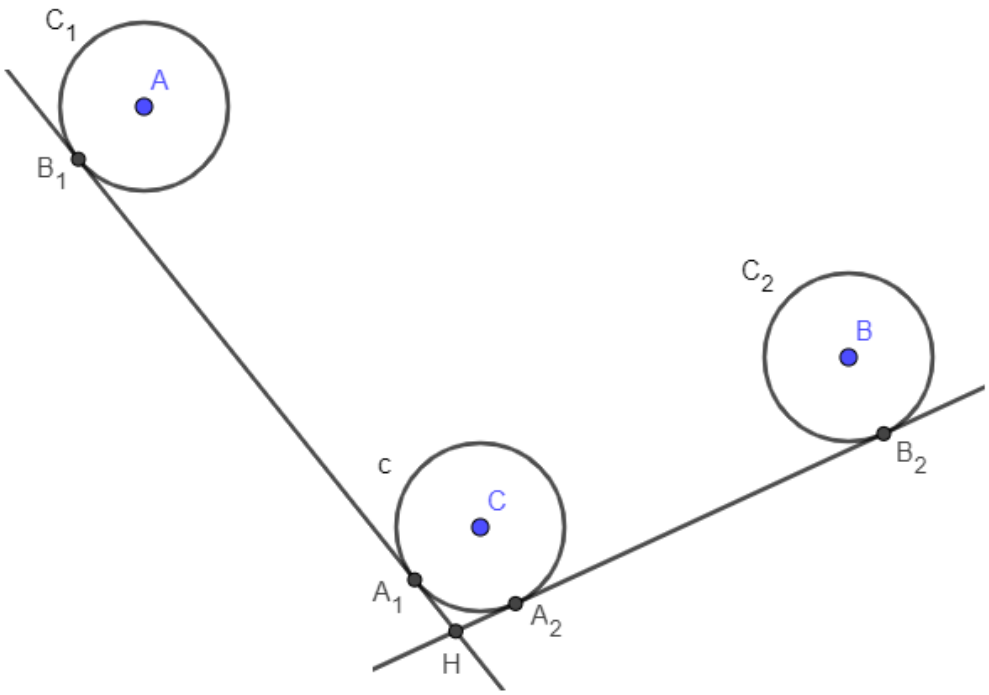


Figura 2.1: Rappresentazione grafica di supporto alla dimostrazione geometrica

2.2 Modelli matematici

Una volta osservate le caratteristiche del problema proponiamo alcuni modelli di programmazione non lineare per provare a risolverlo. In particolare saranno mostrati due differenti approcci. In primo luogo saranno proposti dei modelli integrati che tentano di risolvere il problema nella sua totalità tramite un solutore numerico, successivamente saranno invece presi in considerazione dei modelli che invece suddividono il problema nella sua parte discreta di scelta del tipo di configurazione e nei vari sottoproblemi continui distinti. Nello specifico i modelli integrati sono semplicemente dei modelli di programmazione non lineare che vengono dati in pasto ad un solutore che cerca di estrapolarne l'ottimo. Come però scopriremo, adottando questo approccio si va incontro ad alcuni problemi che ci suggeriranno quindi di sperimentare un'altra tecnica.

2.3 Modelli integrati

Presentiamo due modelli di programmazione non lineare che si prestano alla rappresentazione del problema di interpolazione di Markov-Dubins. Le particolari caratteristiche geometriche dei cammini ottimi ci permettono, grazie all'utilizzo di funzioni goniometriche, di costruire modelli intuitivi ed allo stesso tempo eleganti del problema.

2.3.1 Primo modello MINLP/PNL

Il primo modello sfrutta le caratteristiche della soluzione ottima analizzate nei paragrafi precedenti e fa uso della funzione *sinc* che verrà introdotta a breve. L'idea è quella di essere in grado di rappresentare tutti i tratti di cui abbiamo bisogno mediante un unico tipo di funzione. Dal Teorema 2.1.1 sappiamo che tra due punti consecutivi c'è un tratto del tipo CCC, CLC oppure un loro sottoinsieme; serve quindi un modo intelligente per rappresentare matematicamente ognuno di questi tratti.

Definizione 2.3.1. *Chiamo sinc:* $\mathbb{R} \to \mathbb{R}$ *la funzione definita come segue,* $sinc(x) := \frac{sin(x)}{x}$ *per* $x \neq 0$ *e* $sinc(0) = 1$.

Osservazione: La funzione *sinc* è semplicemente il prolungamento con continuità della funzione $\frac{sin(x)}{x}$
Dalle classiche formule di trigonometria otteniamo le seguenti relazioni:

$$\frac{sin\,(x)}{x} = cos\left(\frac{x}{2}\right) sinc\left(\frac{x}{2}\right) \quad e \quad \frac{1 - cos\,(x)}{x} = sin\left(\frac{x}{2}\right) sinc\left(\frac{x}{2}\right)$$

Siamo ora in grado di scrivere una parametrizzazione che rappresenti sia i tratti rettilinei che gli archi di circonferenza.

Proposizione 2.3.1. *L'equazione parametrica di un segmento rettilineo o di un arco di circonferenza che inizia nel punto* (x_0, y_0) *con angolo iniziale* θ_0 *e con curvatura* κ *(*$\kappa = 0$ *nel caso di segmento rettilineo) e lunghezza* s *è dato da:*
$x(\ell) = x_0 + \ell sinc(\frac{\kappa\ell}{2}) cos(\theta_0 + \frac{\kappa\ell}{2}) \equiv x_0 + f(\ell, \kappa, \theta_0)$
$y(\ell) = y_0 + \ell sinc(\frac{\kappa\ell}{2}) sin(\theta_0 + \frac{\kappa\ell}{2}) \equiv y_0 + g(\ell, \kappa, \theta_0)$
dove $\ell \in [0, s]$.

Dimostrazione. Si veda [10] □

Osservazione: Indichiamo con κ la curvatura che si ottiene come reciproco del raggio di curvatura, come già anticipato si pone uguale a 0 nei tratti dritti in cui per definizione il raggio di curvatura è ∞.

Sfruttando le caratteristiche della soluzione ottima appena mostrate, possiamo riscrivere il problema nella notazione della programmazione non lineare. Quindi, tenendo conto che tra due punti consecutivi ci sono tre tratti (nel caso in cui siano meno si considera la presenza di tratti con lunghezza nulla) e sfruttando la parametrizzazione della Proposizione 2.3.1 otteniamo che il j-esimo tratto del cammino di Markov-Dubins ottimale è vincolato alle seguenti equazioni:

$$x_{j+1} = x_j + f(s_{j,0}, \kappa\sigma_{j,0}, \theta_j) + f(s_{j,1}, \kappa\sigma_{j,1}, \theta_j + \kappa\sigma_{j,0}s_{j,0}) + f(s_{j,2}, \kappa\sigma_{j,2}, \theta_j + \kappa\sigma_{j,0}s_{j,0} + \kappa\sigma_{j,1}s_{j,1})$$

$$y_{j+1} = y_j + g(s_{j,0}, \kappa\sigma_{j,0}, \theta_j) + g(s_{j,1}, \kappa\sigma_{j,1}, \theta_j + \kappa\sigma_{j,0}s_{j,0}) + g(s_{j,2}, \kappa\sigma_{j,2}, \theta_j + \kappa\sigma_{j,0}s_{j,0} + \kappa\sigma_{j,1}s_{j,1})$$

$$\theta_{j+1} = \theta_j + \kappa\sigma_{j,0}s_{j,0} + \kappa\sigma_{j,1}s_{j,1} + \kappa\sigma_{j,2}s_{j,2}$$

Osservazione: L'ultima delle tre equazioni va intesa modulo 2π e l'uguaglianza viene ottenuta uguagliando seno e coseno di entrambi i membri (si veda [18]).

Considerando che il cammino di Markov-Dubins ottimo ha curvatura nell'insieme $\{-\kappa, 0, \kappa\}$ sono state introdotte le variabili intere $\sigma_{j,i} \in \{-1, 0, 1\}$ che vengono appunto moltiplicate al valore della curvatura massima κ di conseguenza il valore $\kappa\sigma_{j,i}$ rappresenta la curvatura dell'i-esimo tratto compreso tra il punto u_j e u_{j+1}. A questo punto è possibile formulare il problema come **MINLP**:

$$
\begin{cases}
min & \displaystyle\sum_{j=0}^{N-1} s_{j,0} + s_{j,1} + s_{j,2} & \\
s.t. & x_{j+1} = x_j + f(s_{j,0}, \kappa\sigma_{j,0}, \theta_j) + f(s_{j,1}, \kappa\sigma_{j,1}, \theta_j + \kappa\sigma_{j,0}s_{j,0}) + & \\
 & \qquad + f(s_{j,2}, \kappa\sigma_{j,2}, \theta_j + \kappa\sigma_{j,0}s_{j,0} + \kappa\sigma_{j,1}s_{j,1}) & \\
 & y_{j+1} = y_j + g(s_{j,0}, \kappa\sigma_{j,0}, \theta_j) + g(s_{j,1}, \kappa\sigma_{j,1}, \theta_j + \kappa\sigma_{j,0}s_{j,0}) + & \\
 & \qquad + g(s_{j,2}, \kappa\sigma_{j,2}, \theta_j + \kappa\sigma_{j,0}s_{j,0} + \kappa\sigma_{j,1}s_{j,1}) & \\
 & \theta_{j+1} = \theta_j + \kappa\sigma_{j,0}s_{j,0} + \kappa\sigma_{j,1}s_{j,1} + \kappa\sigma_{j,2}s_{j,2} & j = 0, ..., N-1 \\
 & (x_j, y_j) = u_j & j = 0, ..., N \\
 & \theta_0, \theta_N \quad fissati & \\
 & \sigma_{j,i} = \{-1, 0, 1\} & j = 0, ..., N-1, \quad i = 0, 1, 2 \\
 & s_{j,i} \geq 0, & j = 0, ..., N-1, \quad i = 0, 1, 2 \\
 & \theta_j \in \mathbb{R} & j = 1, ..., N-1
\end{cases}
$$

dove ricordiamo che $f(\ell, \kappa, \theta_0) = \ell sinc(\frac{\kappa\ell}{2}) cos(\theta_0 + \frac{\kappa\ell}{2})$ e
$g(\ell, \kappa, \theta_0) = \ell sinc(\frac{\kappa\ell}{2}) sin(\theta_0 + \frac{\kappa\ell}{2})$.

Osservazione: La terza uguaglianza del sistema ci fornisce la continuità C^1 del cammino.

Osservazione: Nel caso in cui non siano forniti gli angoli di partenza e/o arrivo è evidente come generalizzare la formulazione.

Data l'elevata complessità computazionale del problema, potrebbe essere vantaggioso rilassare i vincoli di integrità su $\sigma_{j,i}$ ponendo $-1 \leq \sigma_{j,i} \leq 1$ ottenendo così la formulazione di un **PNL**.

Osservazione: I solutori di **PNL** o **MINLP** richiedono in input un vettore iniziale di dati da cui partire per ricerca dell'ottimo, per questo motivo potrebbe essere utile ricavare una soluzione ammissibile (non ottima) mediante un metodo euristico. In alternativa è possibile utilizzare un metodo multistart, cioè lasciare al solutore la possibilità di generare automaticamente una serie di valori di partenza con cui operare.

2.3.2 Secondo modello PNL

Proponiamo ora un secondo modello per l'individuazione della soluzione ottima del problema (Pc). (Per ulteriori approfondimenti si veda [18].) Per ogni tratto i congiungente due punti consecutivi, con $i = 1, ..., N$, si consideri $\epsilon_j^i := t_j^i - t_{j-1}^i$ per $j = 1, ..., 5$ che rappresenta la lunghezza del j-esimo sottoarco del tratto i dove t_j^i indica l'istante in cui avviene il cambio tra i diversi tipi di archi. Chiaramente $t_0^1 := 0$ e $t_5^N := t_N$. Ora, come conseguenza del Teorema 2.1.1 il cammino ottimo può essere rappresentato come concatenazione di tratti del tipo $L_{\epsilon_1^i} R_{\epsilon_2^i} S_{\epsilon_3^i} L_{\epsilon_4^i} R_{\epsilon_5^i}$ dove si ricorda che nel caso ottimo al più tre dei cinque pezzi possono risultare diversi da zero. Considerando le soluzioni delle equazioni differenziali del problema (Pc) per $t_{j-1}^i \leq t \leq t_j^i$ con $i = 1, ..., N$

$$\theta(t) = \theta(t_{j-1}^i) + u(t)(t - t_{j-1}^i) \qquad j = 1, ..., 5$$

$$x(t) = \begin{cases} x(t_{j-1}^i) + (sin\theta(t) - sin\theta(t_{j-1}^i))/u(t) & j = 1, 2, 4, 5 \\ x(t_{j-1}^i) + cos\theta(t)(t - t_{j-1}^i) & j = 3 \end{cases}$$

$$y(t) = \begin{cases} y(t_{j-1}^i) - (cos\theta(t) - cos\theta(t_{j-1}^i))/u(t) & j = 1, 2, 4, 5 \\ y(t_{j-1}^i) + sin\theta(t)(t - t_{j-1}^i) & j = 3 \end{cases}$$

dove

$$u(t) = \begin{cases} \kappa & j = 1, 4 \\ -\kappa & j = 2, 5 \\ 0 & j = 3 \end{cases}$$

il problema può essere riscritto come segue:

$$\begin{cases} min & t_N = \sum_{i=1}^{N} \sum_{j=1}^{5} \epsilon_j^i \\ s.t. & x_{i-1} - x_i + \frac{1}{\kappa}(-sin\theta_0^i + 2sin\theta_1^i - 2sin\theta_2^i + 2sin\theta_4^i - sin\theta_5^i) + \epsilon_3^i cos\theta_2^i = 0 \\ & y_{i-1} - y_i + \frac{1}{\kappa}(cos\theta_0^i - 2cos\theta_1^i + 2cos\theta_2^i - 2cos\theta_4^i + cos\theta_5^i) + \epsilon_3^i sin\theta_2^i = 0 \\ & x_0, y_0, x_N, y_N \ \ fissati \\ & \theta_0^1 = \theta_0, \quad sin\theta_5^N = sin\theta_N, \quad cos\theta_5^N = cos\theta_N \\ & \theta_0^{i+1} = \theta_5^i \qquad\qquad i = 1, ..., N-1 \\ & \epsilon_j^i \geq 0 \qquad\qquad i = 1, ..., N, j = 1, ..., 5 \end{cases}$$

dove $\theta_1^i = \theta_0^i + \kappa\epsilon_1^i, \qquad \theta_2^i = \theta_1^i - \kappa\epsilon_2^i, \qquad \theta_4^i = \theta_2^i + \kappa\epsilon_4^i, \qquad \theta_5^i = \theta_4^i - \kappa\epsilon_5^i.$

2.3.3 Risultati numerici

Nei test numerici dei problemi di programmazione non lineare affrontati nel corso della tesi è stato necessario fare ricorso a solutori. In particolare è stato utilizzato il software di ottimizzazione non lineare **Knitro** nella sua versione 12.4.0 [19] appoggiandosi sia a Matlab che ad AMPL.
La prima problematica da affrontare deriva dal fatto che facendo uso di solutori non lineari, non è garantita la convergenza ad un ottimo globale. Quello che spesso succede è che il risultato ottenuto si limita ad essere un ottimo locale.

Definizione 2.3.2. *Chiamo* ***ottimo locale*** *una soluzione che non può essere migliorata mediante piccole perturbazioni negli angoli.*

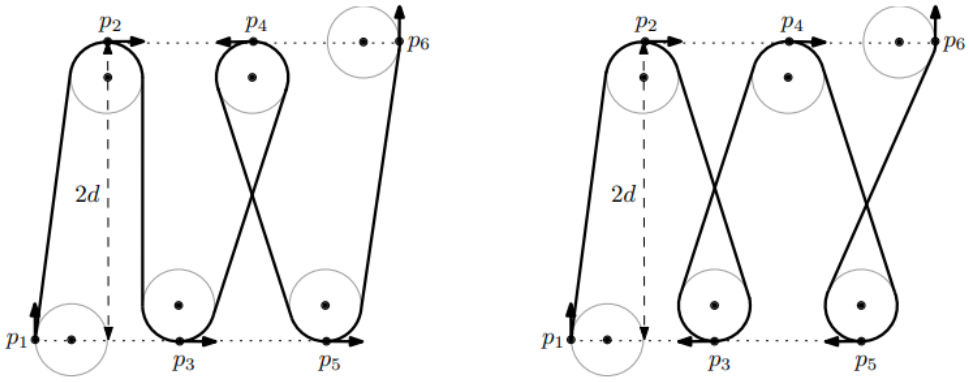
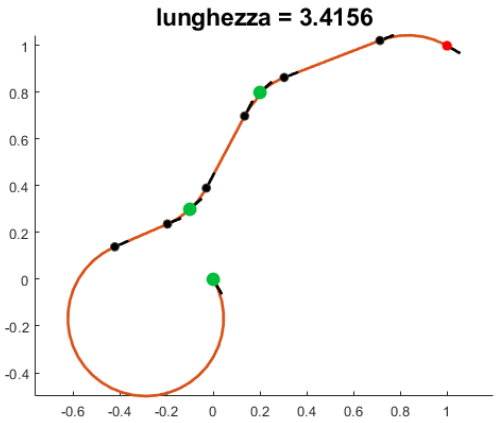


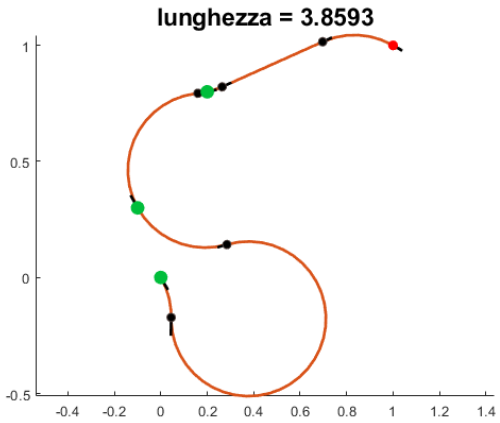
Figura 2.2: Esempi di ottimi locali distinti tratti da [12]

Nei problemi di dimensioni non banali le soluzioni locali crescono in maniera esponenziale: consideriamo per esempio la sequenza di punti $\{(2i, (-1)^i d) | i = 1, ..., n\}$ in modo che d sia abbastanza grande, la sequenza di angoli $(\theta_2, ..., \theta_{n-1}) \in \{0, \pi\}^{n-2}$ definisce 2^{n-2} cammini che approssimano altrettanti distinti ottimi locali. (Si veda Figura 2.2). Questa osservazione rende pertanto i problemi in un certo senso mal condizionati, infatti partendo da un dato iniziale generico non abbiamo nessuna garanzia di convergere alla soluzione ottima. Una prima proposta per ovviare a questa problematica è quella di sfruttare la Proposizione 2.1.2 imponendo degli ulteriori vincoli al problema. Alternativamente, per evitare di appesantire il carico computazionale è possibile limitarsi a verificare che la soluzione trovata soddisfi la sopracitata proprietà e in caso negativo catalogarla come ottimo locale. Come già osservato in precedenza, i solutori non lineari, e Knitro di conseguenza, chiedono in input un dato iniziale da cui iniziare la ricerca dell'ottimo. La cosa migliore sarebbe fornire al solutore una soluzione ammissibile o quasi non lontana dall'ottimo globale. Tale richiesta necessiterebbe di un metodo alternativo per generare una soluzione adeguata da cui partire. Per semplicità tuttavia, almeno in una prima fase, preferiamo non fornire un dato iniziale, lasciando a Knitro la possibilità di assegnarsi automaticamente un punto di partenza. Col fine di limitare i possibili malfunzionamenti di questa tecnica, viene abilitato un approccio multistart, cioè vengono generati numerosi (200 di default) dati differenti con cui partire nella ricerca dell'ottimo. Ci si aspetta che tra i dati iniziali ce ne sia almeno uno che faccia convergere all'ottimo globale.

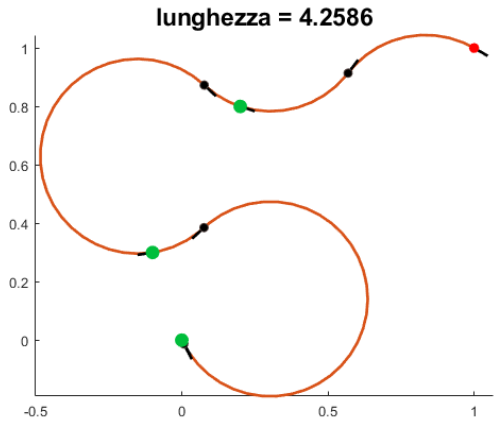
Dei due modelli abbiamo testato soprattutto il secondo, in grado di offrire i migliori risultati, che tuttavia non si possono considerare del tutto soddisfacenti. Il problema numerico è molto complesso e il solutore non lineare ha la pretesa di risolverlo senza avere molte informazioni su di esso. Nonostante i problemi più semplici come per esempio il Problema 1 in Appendice A vengano risolti all'ottimo senza difficoltà, quando il numero di punti cresce il problema degli ottimi locali si rivela spinoso. Affidarsi al solo approccio multistart potrebbe non essere sufficiente per scovare l'ottimo globale, anzi, al crescere dei punti da raggiungere, capita che nemmeno un ottimo locale venga raggiunto e la ricerca si fermi senza trovare soluzioni ammissibili. Qui alcuni esempi di ottimi locali che si possono ottenere dalla risoluzione del Problema 2 in Appendice A.



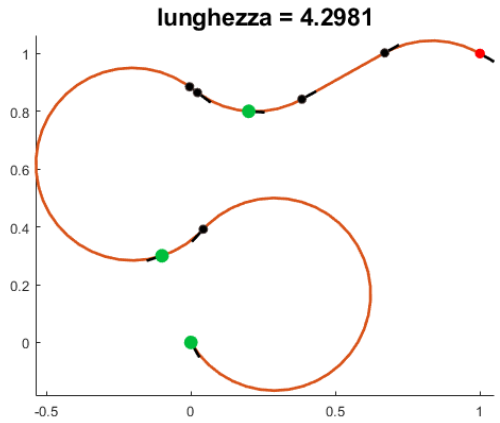
RLR|LSR|RSR ≡ RSLSRSR



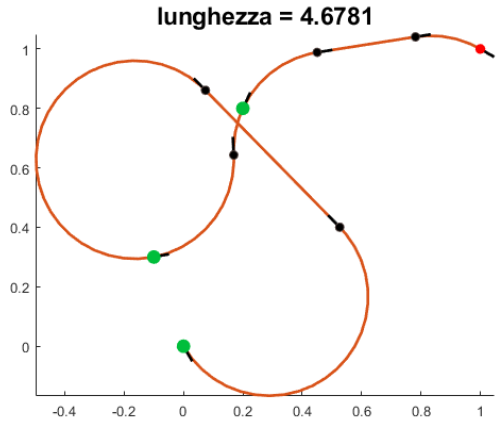
RLR|RL|LSR ≡ RLRLSR



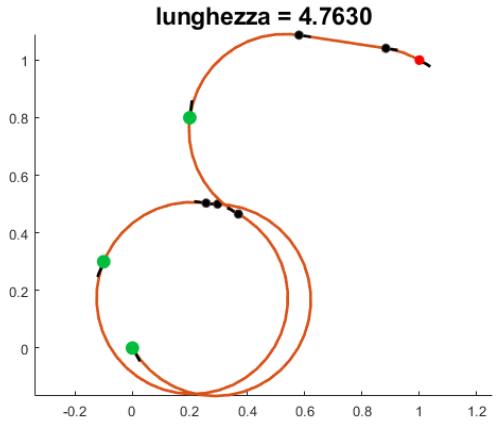
RLR|RL|LR ≡ RLRLR



LR|RSL|LSR ≡ LRSLSR



LSL|LR|RSR ≡ LSLRSR



LRL|LR|RSR ≡ LRLRSR

2.4 Nuove strategie

Nonostante siano state individuate alcune caratteristiche del cammino ottimo e che alcune di esse sono delle condizioni necessarie di facile verifica, non è tuttavia semplice risolvere il problema come invece succede nel caso $N = 1$. Le strade da seguire possono essere molteplici [10]: è possibile esplorare il percorso ottimale tramite un'euristica, è possibile altresì procedere con una discretizzazione del problema oppure provare a risolverlo, adoperando i dovuti accorgimenti, sotto forma di problema di ottimizzazione non lineare o ottimizzazione non lineare intera. Poiché la velocità è costante, trovare il cammino minimo corrisponde a trovare la corretta orientazione di passaggio θ_i per ogni punto u_i. Se indichiamo con $d(\theta_i, \theta_{i+1})$ la distanza ottima nel senso di Dubins tra due punti successivi u_i e u_{i+1} fissati gli angoli θ_i, θ_{i+1}, il cammino ottimo è dato da $X^* = \min_{\theta_1,\dots,\theta_{N-1}} \sum_{i=0}^{N-1} d(\theta_i, \theta_{i+1})$. Ora è chiaro che limitarsi ai modelli non lineari, almeno così come presentati finora, non possa essere sufficiente per la risoluzione di un problema generico, poiché la poca robustezza su istanze medio-grandi non fornisce l'affidabilità auspicata. Le difficoltà sorgono dal fatto che si prova a risolvere il problema "a scatola chiusa" senza ricavare delle informazioni sul tipo di cammino da seguire. Nel seguito del lavoro si procederà pertanto provando a risolvere il problema scindendo la sua parte discreta da quella continua. Questa suddivisione in parte continua e discreta sorge abbastanza naturale infatti può essere intelligente distinguere il problema discreto di scegliere il tipo di curva da seguire in ogni punto (destra o sinistra) dal problema continuo di ottimizzare il cammino una volta fissate le configurazioni di passaggio. Di conseguenza il sottoproblema continuo è un PNL le cui variabili sono gli angoli di passaggio θ_i e va risolto, in maniera analoga ai modelli presentati in precedenza, con l'ausilio di un solutore software. Il problema discreto invece ha come variabili le orientazioni delle varie curve (che indicheremo con σ_i) e come vedremo sarà risolto mediante enumerazione implicita.

2.4.1 Cammino ottimo nota la configurazione da seguire

Consideriamo ora il caso in cui il tipo di cammino ottimo da seguire sia noto e il raggio di curvatura unitario. In analogia con quanto fatto nel Paragrafo 1.2.4 consideriamo gli operatori R_θ, L_θ, S_v [7] che rappresentano rispettivamente un tratto curvo a destra, uno a sinistra ed un tratto rettilineo, che a partire dalla generica configurazione (x, y, φ) mandano nella corrispondente immagine:

- $\mathrm{L}_\theta(x, y, \varphi) = (x + sin(\varphi + \theta) - sin(\varphi), y - cos(\varphi + \theta) + cos(\varphi), \varphi + \theta)$
- $\mathrm{R}_\theta(x, y, \varphi) = (x - sin(\varphi - \theta) + sin(\varphi), y + cos(\varphi - \theta) - cos(\varphi), \varphi - \theta)$
- $\mathrm{S}_v(x, y, \varphi) = (x + v \cdot cos(\varphi), y + v \cdot sin(\varphi), \varphi)$

v e θ rappresentano rispettivamente la lunghezza dei tratti rettilinei e di quelli curvi, inoltre poiché il raggio è unitario θ rappresenta anche l'angolo percorso. Si consideri ora un generico tratto del tipo LSR che dalla configurazione (x_0, y_0, θ_0) giunge in (x_1, y_1, θ_1) , chiaramente $R_q(S_p(L_t(x_0, y_0, \theta_0))) = (x_1, y_1, \theta_1)$ dove la lunghezza totale del cammino si ottiene sommando le lunghezze dei singoli tratti t, p, q. Applicando gli operatori appena mostrati otteniamo le seguenti equazioni:

$$\begin{cases} x_0 + sin(\theta_0 + t) - sin(\theta_0) + p \cdot cos(\theta_0 + t) - sin(\theta_0 + t - q) + sin(\theta_0 + t) = x_1 \\ y_0 - cos(\theta_0 + t) + cos(\theta_0) + p \cdot sin(\theta_0 + t) + cos(\theta_0 + t - q) - cos(\theta_0 + t) = y_1 \\ \theta_0 + t - q = \theta_1 (mod\ 2\pi) \end{cases}$$

le cui soluzioni sono

$$\begin{cases} p = \sqrt{\tilde{d}} \\ t = (A - B - \theta_0)(mod\ 2\pi) \\ q = (A - B - \theta_1)(mod\ 2\pi) \end{cases}$$

dove

$$\mathrm{A} = \begin{cases} \theta + \pi & y_1 - y_0 - cos(\theta_0) - cos(\theta_1) > 0 \\ \theta & y_1 - y_0 - cos(\theta_0) - cos(\theta_1) < 0 \end{cases} \quad , \quad \theta = arctan(-\tfrac{x_1 - x_0 + sin(\theta_0) + sin(\theta_1)}{y_1 - y_0 - cos(\theta_0) - cos(\theta_1)}),$$

$B = \sqrt{\frac{\tilde{d}}{2}}, \quad \tilde{d} = d^2 - 2 + 2cos(\theta_0 - \theta_1) + 2(x_1 - x_0)(sin(\theta_0) + sin(\theta_1)) +$
$- 2(y_1 - y_0)(cos(\theta_0) + cos(\theta_1))$ con $d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$ distanza euclidea.
In maniera del tutto analoga possiamo ricavare le equazioni di tutte le tipologie di cammino con le rispettive soluzioni. Pertanto, conoscendo le configurazioni del cammino ottimo, il problema di minimo può essere riscritto facilmente per via analitica come segue.

$$\begin{cases} min & \sum_{i=1}^{N} p_i + t_i + q_i \\ s.t. & p_i, t_i, q_i \text{ lunghezze dei tre tratti che congiungono il punto } i-1 \text{ al punto } i.\ \forall i = 1, ..., N \\ & \text{le varie lunghezze } p_i, t_i, q_i \text{ sono le soluzioni dei rispettivi sistemi di equazioni associati} \\ & \text{alle particolari tipologie di percorso e sono funzioni delle variabili angolo } \theta_i \end{cases}$$

Per via delle numerose equazioni goniometriche non è per nulla banale ricavare l'ottimo in maniera esplicita. È conveniente, come anticipato, affidarsi nuovamente ad un solutore numerico.

2.4.2 Terzo modello PNL

Consideriamo ora una sequenza di punti per la quale il cammino ottimo è formato da una concatenazione di tratti di tipo CSC[2]. Supponiamo inoltre di conoscere l'orientazione di passaggio (R o L) lungo i tratti curvi. Siamo così in grado di costruire un modello non lineare più semplice e con meno variabili rispetto a quelli visti nel paragrafo precedente. Indichiamo con u_i per $i = 0, ..., N$ i punti da visitare e con θ_i per $i = 0, ..., N$ gli angoli di passaggio ad ogni punto. Gli angoli positivi si misurano in senso antiorario a partire dal semiasse positivo delle ascisse, sono invece negativi in senso orario. Indichiamo poi con α_i l'angolo della direzione dal punto u_i al centro della circonferenza su cui è posizionato il relativo tratto curvo e con γ_i l'angolo percorso durante il singolo tratto curvo. In generale vale $\gamma_i \in [0, 2\pi)$ se la curva è di tipo L mentre $\gamma_i \in (-2\pi, 0]$ se la curva è R. Considerando che il raggio delle circonferenze è unitario e tenendo a mente la Proposizione 2.1.2 sappiamo che $|\gamma_i|$ rappresenta la lunghezza del tratto curvo sia che precede sia che segue il punto u_i con ovvie eccezioni per u_0 e u_N.
Valgono le seguenti relazioni:

- $\theta_i + \gamma_i \overset{2\pi}{=} \theta_{i+1} - \gamma_{i+1} \qquad \forall i = 0, ..., N-1$
- $\theta_i \overset{2\pi}{=} \alpha_i - \dfrac{\pi}{2} \qquad \forall i = 0, ..., N \ \ t.c.$ la curva è sinistra.
- $\theta_i \overset{2\pi}{=} \alpha_i + \dfrac{\pi}{2} \qquad \forall i = 0, ..., N \ \ t.c.$ la curva è destra.

Per evitare le uguaglianze modulo 2π possiamo sostituire gli angoli con le coppie di valori seno e coseno ottenendo:

1. $\begin{cases} sin(\theta_i + \gamma_i) = sin(\theta_{i+1} - \gamma_{i+1}) \\ cos(\theta_i + \gamma_i) = cos(\theta_{i+1} - \gamma_{i+1}) \end{cases} \qquad \forall i = 0, ..., N-1$
2. $\begin{cases} sin(\theta_i) = -cos(\alpha_i) \\ cos(\theta_i) = +sin(\alpha_i) \end{cases} \qquad \forall i = 0, ..., N \ \ t.c.$ la curva è sinistra.
3. $\begin{cases} sin(\theta_i) = +cos(\alpha_i) \\ cos(\theta_i) = -sin(\alpha_i) \end{cases} \qquad \forall i = 0, ..., N \ \ t.c.$ la curva è destra.

Prima di delineare il modello facciamo ancora un paio di considerazioni. Nella funzione obiettivo da minimizzare abbiamo bisogno di sapere la lunghezza dei vari tratti, oltre a quelli curvi di lunghezza $|\gamma_i|$, ci serve sapere la lunghezza anche di quelli rettilinei. Se due curve consecutive sono dello stesso tipo (entrambe destre

[2]Come abbiamo già osservato, una condizione sufficiente per cui ciò accada è che i punti consecutivi distino l'uno dall'altro una misura superiore a $4r$

o entrambe sinistre), la lunghezza del tratto rettilineo tra di esse coincide con la distanza tra i centri delle circonferenze su cui sono posti i tratti curvi. Se invece le curve sono in direzioni opposte la lunghezza del tratto rettilineo coincide con il cateto di un triangolo rettangolo che ha per ipotenusa la distanza tra i centri delle circonferenze su cui sono posti i tratti curvi e come altro cateto 2, cioè il diametro delle circonferenze di raggio unitario. Quindi nel primo caso la lunghezza del tratto rettilineo tra u_i e u_{i+1} è

$$S_{i,i+1} = \sqrt{(x_{i+1} - x_i + cos(\alpha_{i+1}) - cos(\alpha_i))^2 + (y_{i+1} - y_i + sin(\alpha_{i+1}) - sin(\alpha_i))^2}$$

mentre nel secondo caso

$$S_{i,i+1} = \sqrt{(x_{i+1} - x_i + cos(\alpha_{i+1}) - cos(\alpha_i))^2 + (y_{i+1} - y_i + sin(\alpha_{i+1}) - sin(\alpha_i))^2 - 4}.$$

Abbiamo così tutti gli ingredienti per formalizzare il modello matematico.

$$\begin{cases} min & |\gamma_0| + \left(\sum_{i=1}^{N-1} 2 \cdot |\gamma_i| \right) + |\gamma_N| + \sum_{i=0}^{N-1} S_{i,i+1} \\ s.t. & \forall i = 0, ..., N-1 \\ & sin(\theta_i + \gamma_i) = sin(\theta_{i+1} - \gamma_{i+1}) \\ & cos(\theta_i + \gamma_i) = cos(\theta_{i+1} - \gamma_{i+1}) \\ & \sigma_j(sin(\gamma_i) - sin(\gamma_i + \theta_i)) + S_{i,i+1} \cdot cos(\gamma_i + \theta_i) + \\ & \qquad + \sigma_j(sin(\gamma_{i+1} - \theta_{i+1}) - sin(\gamma_{i+1})) + x_i - x_{i+1} = 0 \\ & \sigma_j(-cos(\gamma_i) + cos(\gamma_i + \theta_i)) + S_{i,i+1} \cdot sin(\gamma_i + \theta_i) + \\ & \qquad + \sigma_j(-cos(\gamma_{i+1} - \theta_{i+1}) + cos(\gamma_{i+1})) + y_i - y_{i+1} = 0 \\ & \text{dove } \sigma_j = 1 \text{ se la curva di riferimento è destra e } \sigma_j = -1 \text{ nel caso in cui è sinistra.} \end{cases}$$

Facendo uso delle note formule trigonometriche di addizione e sottrazione

$$cos(\alpha \pm \beta) = cos(\alpha)cos(\beta) \mp sin(\alpha)sin(\beta) \qquad sin(\alpha \pm \beta) = sin(\alpha)cos(\beta) \pm cos(\alpha)sin(\beta)$$

e delle uguaglianze (2) e (3) è possibile riscrivere il problema non lineare appena enunciato nelle variabili γ_i, $sin(\theta_i)$, $cos(\theta_i)$ $\forall i = 0, ..., N$ dove i valori di seno e coseno di θ_0 e θ_N potrebbero essere noti. Rispetto ai modelli presentati nei paragrafi precedenti, questo presenta un numero minore di variabili. Ricordiamo però che tale modello è valido a patto di essere a conoscenza di alcune caratteristiche del sistema, cioè che il cammino ottimo è una concatenazione di percorsi del tipo CSC dei quali sono noti i tipi di curve (R o L), condizioni spesso incognite in un problema generico. Testiamo ora questa formulazione al calcolatore utilizzando ancora una volta **Knitro** con il linguaggio di programmazione AMPL. Facendo uso nuovamente di un approccio multistart il modello si rivela efficace sui Problemi 1, 6 e 7 in Appendice A. Il vantaggio del modello presentato ora è quello di tenere contenuto il numero di variabili, inoltre è possibile fissare dei bound superiori ed inferiori migliori rispetto ai modelli precedenti favorendo appunto un approccio senza valori di partenza delle variabili. È opportuno evidenziare inoltre che il fatto di fissare le configurazioni del cammino permette di evitare le spiacevoli situazioni in cui l'ottimo individuato è in realtà un ottimo locale. Nasce pertanto la necessità di costruire un algoritmo in grado di risolvere il sottoproblema discreto ed individuare la sequenza ottima delle curve.

2.4.3 Modelli del sottoproblema continuo

Prima di costruire l'algoritmo di programmazione dinamica in grado di combinare i sottoproblema discreto con quello continuo proponiamo due modelli non lineari adatti alla rappresentazione del sottoproblema continuo. Facciamo riferimento a modelli in cui la distanza tra punti consecutivi è maggiore di $4r$ nei quali tra ogni due punti c'è un tratto del tipo CSC.

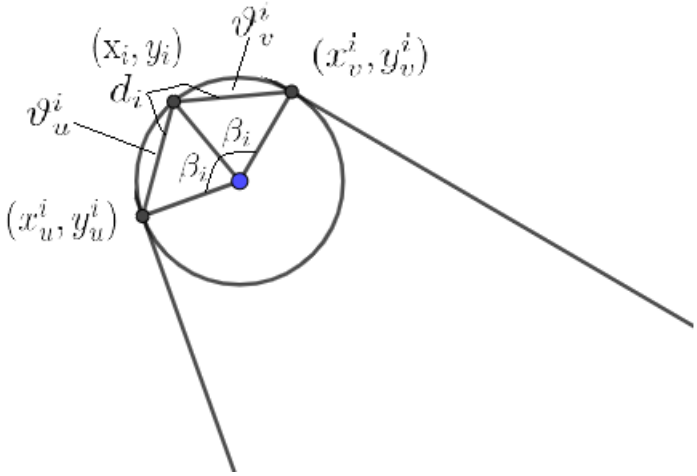
Quarto modello PNL

Parametri noti:

- x_i, y_i per $i = 0, ..., N$ che rappresentano le coordinate del punti di passaggio.
- $\sigma_i \in \{-1, 1\}$ per $i = 1, ..., N$ che rappresenta l'orientamento a destra o a sinistra della curva (R o L) nel punto $i-$esimo.
- r raggio di curvatura.

Variabili:

- β_i per $i = 1, ..., N$ che rappresenta l'ampiezza di mezzo arco di circonferenza, cioè dal punto di passaggio $i-$esimo al precedente o successivo tratto rettilineo. Nel caso in cui la direzione iniziale e finale siano libere, quando possibile, $\beta_0 = \beta_N = 0$. Vale inoltre $\pi \cdot (\sigma_i - 1)/2 \leq \beta_i \leq \pi \cdot (1 + \sigma_i)/2$
- x_u^i, y_u^i per $i = 1, ..., N$ che rappresentano le coordinate del punto di scambio fra tratto rettilineo e tratto curvo precedente all'$i-$esimo punto, cioè il punto di tangenza del tratto rettilineo precedente al punto i con la circonferenza su cui poggia il tratto curvo.
- x_v^i, y_v^i per $i = 0, ..., N-1$ che rappresentano le coordinate del punto di scambio fra tratto rettilineo e tratto curvo successivo all'$i-$esimo punto, cioè il punto di tangenza del tratto rettilineo successivo al punto i con la circonferenza su cui poggia il tratto curvo.
- θ_u^i per $i = 1, ..., N$ che rappresenta la direzione del segmento che va dal punto (x_u^i, y_u^i) al punto (x_i, y_i).
- θ_v^i per $i = 0, ..., N-1$ che rappresenta la direzione del segmento che va dal punto (x_i, y_i) al punto (x_v^i, y_v^i).
- l_i per $i = 1, ..., N$ che rappresenta la lunghezza del tratto rettilineo che va dal punto (x_v^{i-1}, y_v^{i-1}) al punto (x_u^i, y_u^i).
- a_i per $i = 1, ..., N$ che rappresenta la pendenza del tratto rettilineo l_i
- d_i per $i = 0, ..., N$ che rappresenta la lunghezza della corda che va dal punto (x_u^i, y_u^i) al punto (x_i, y_i) e la corda congruente che va dal punto (x_i, y_i) al punto (x_v^i, y_v^i)



Modello:

$$\begin{cases} x_u^i = x_v^{i-1} + l_i * cos(a_i) & i = 1, ..., N \\ y_u^i = y_v^{i-1} + l_i * sin(a_i) & i = 1, ..., N \\ x_u^i + d_i * cos(\theta_u^i) = x_i & i = 1, ..., N \\ y_u^i + d_i * sin(\theta_u^i) = y_i & i = 1, ..., N \\ x_v^i - d_i * cos(\theta_v^i) = x_i & i = 0, ..., N-1 \\ y_v^i - d_i * sin(\theta_v^i) = y_i & i = 0, ..., N-1 \\ (d_i)^2 = 2r^2 * (1 - cos(\beta_i)) & i = 0, ..., N \\ a_i + \beta_i/2 = \theta_u^i & i = 1, ..., N \\ \theta_u^i + \beta_i = \theta_v^i & i = 1, ..., N-1 \\ \theta_v^i + \beta_i/2 = a_{i+1} & i = 0, ..., N-1 \\ \beta_0 = 0 \quad x_v^0 = x_0, \quad y_v^0 = y_0 \end{cases}$$

L'ultima riga rappresenta le condizioni al bordo per il tratto iniziale e può essere facilmente adattata nel caso in cui la direzione sia fissata. In tal caso, o nel caso di direzione di arrivo fissata, è sufficiente imporre $\theta_v^0 - \beta_0/2 = \theta_0$ e $\theta_u^N + \beta_N/2 = \theta_N$, con θ_0 e θ_N le direzioni rispettivamente di partenza e arrivo del cammino. Quello che si osserva risolvendo il modello, è che quando possibile, anche senza imporre nessun tipo di condizione, $\beta_N = 0, \quad x_u^N = x_N, \quad y_u^N = y_N$. È noto infatti che se le direzioni di partenza e arrivo sono libere la lunghezza degli archi di circonferenza iniziale e finale del cammino risultano essere di lunghezza zero. Se stiamo quindi cercando di risolvere il modello completo con l'intera sequenza di punti è consigliabile imporre le condizioni sul tratto finale in modo da "aiutare" la convergenza del solutore. Nel caso in cui stiamo cercando di risolvere un sottoproblema che termina in un punto u_i con $i \neq N$ è invece necessario rilassare il vincolo. Questo è necessario perché, nonostante nella stragrande maggioranza dei casi il cammino ottimo inizia e finisce con un tratto rettilineo e non necessita di archi di circonferenza, vincolando il cammino a delle determinate configurazioni è possibile che ciò non accada e ci sia bisogno di un arco di circonferenza conclusivo. Si verificano queste eccezioni solamente in alcuni casi "patologici" dove i punti sono posizionati "quasi allineati". È evidente come questo tipo di cammini non possa essere ottimo

se il cammino terminasse in u_i, ma tale configurazione potrebbe poi rivelarsi utile nella prosecuzione del percorso verso il punto u_{i+1}.
Il funzionamento del modello è stato testato sulle istanze del Problema 1, Problema 6 e Problema 7 dell'appendice A fornendo buone performance e giungendo all'ottimo in tempi rapidi. In tali test è noto a priori il vettore σ scelto in maniera tale da ottenere il percorso ottimo. In questi casi, oltre che con **Knitro**, i test sono stati eseguiti anche con il solutore non lineare **SNOPT**.

Quinto modello PNL

Proponiamo un ulteriore modello: i parametri noti sono identici a quelli del modello precedente ma cambiano alcune variabili del problema. Consideriamo senza perdita di generalità $r = 1$.

Variabili:

- β_i per $i = 1, ..., N$ che rappresenta l'ampiezza di mezzo arco di circonferenza, cioè dal punto di passaggio $i-$esimo al precedente o successivo tratto rettilineo. Vale inoltre $\pi \cdot (\sigma_i - 1)/2 \leq \beta_i \leq \pi \cdot (1 + \sigma_i)/2$

- x_u^i, y_u^i per $i = 1, ..., N$ che rappresentano le coordinate del punto di scambio fra tratto rettilineo e tratto curvo precedente all'$i-$esimo punto, cioè il punto di tangenza del tratto rettilineo precedente al punto i con la circonferenza su cui poggia il tratto curvo.

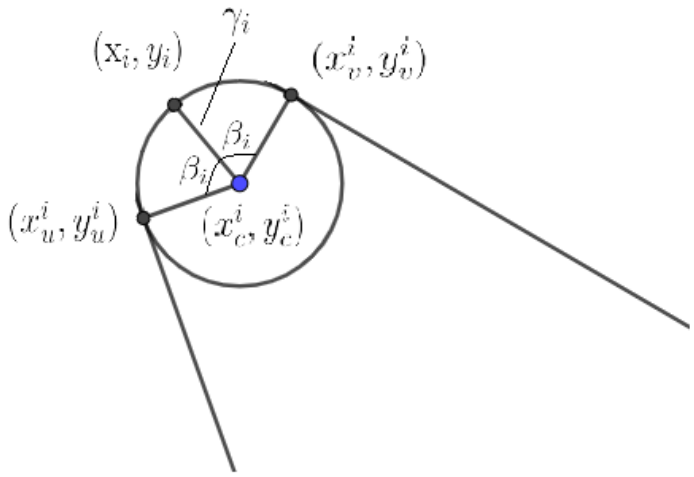
- x_v^i, y_v^i per $i = 1, ..., N-1$ che rappresentano le coordinate del punto di scambio fra tratto rettilineo e tratto curvo successivo all'$i-$esimo punto, cioè il punto di tangenza del tratto rettilineo successivo al punto i con la circonferenza su cui poggia il tratto curvo.

- l_i per $i = 1, ..., N$ che rappresenta la lunghezza del segmento rettilineo che precede il punto (x_i, y_i).

- θ_i per $i = 1, ..., N$ che rappresenta la direzione del segmento rettilineo precedente al punto (x_i, y_i).

- x_c^i, y_c^i per $i = 1, ..., N$ che rappresentano le coordinate del centro della circonferenza su cui poggia il tratto curvo $i-$esimo.

- γ_i per $i = 1, ..., N$ che rappresenta la direzione dal punto (xc_i, yc_i) al punto (x_i, y_i).



Modello:

$$\begin{cases} x_i = x_c^i + cos(\gamma_i) & i = 1, ..., N \\ y_i = y_c^i + sin(\gamma_i) & i = 1, ..., N \\ x_u^i = x_c^i + cos(\gamma_i - \beta_i) & i = 1, ..., N \\ y_u^i = y_c^i + sin(\gamma_i - \beta_i) & i = 1, ..., N \\ x_v^i = x_c^i + cos(\gamma_i + \beta_i) & i = 1, ..., N-1 \\ y_v^i = y_c^i + sin(\gamma_i + \beta_i) & i = 1, ..., N-1 \\ \gamma_i + \sigma_i * pi/2 = \theta_i + \beta_i & i = 1, ..., N \\ \theta_{i+1} = \theta_i + 2 * \beta_i & i = 1, ..., N-1 \\ x_u^1 = x_0 + l_1 * cos(\theta_1) & \\ y_u^1 = y_0 + l_1 * sin(\theta_1) & \\ x_u^i = x_v^{i-1} + l_i * cos(\theta_i) & i = 2, ..., N \\ y_u^i = y_v^{i-1} + l_i * sin(\theta_i) & i = 2, ..., N \end{cases}$$

Questo modello, forse più intuitivo del precedente, permette performance più brillanti ed è in grado di trovare soluzioni a problemi in cui il quarto modello falliva e portava ad inammissibilità. Per quanto resti un modello abbastanza complesso e con la presenza di numerosi vincoli, nei nostri test con un numero non eccessivo di punti da raggiungere ha portato senza eccessive difficoltà all'ottimo. Per fare in modo che la ricerca dell'ottimo sia efficace è necessaria però una buona inizializzazione. Se in istanze piccole e più fortunate lasciare che il solutore inizializzi automaticamente le variabili non crea problemi, non si può dire lo stesso quando il problema inizia a diventare più grande. Prima di fare qualche considerazione sulle inizializzazioni, mostriamo la funzione obiettivo che minimizza la lunghezza totale del cammino. Per evitare le non linearità dovute alla presenza dei valori assoluti, è possibile moltiplicare i β_i con i corrispondenti σ_i rendendo la funzione obiettivo

$$\left(\sum_{i=1}^{N-1} 2 \cdot \sigma_i \cdot \beta_i \right) + \cdot \sigma_N \cdot \beta_N + \sum_{i=1}^{N} l_i$$

2.4.4 Vantaggi e svantaggi

Il principale vantaggio che deriva dall'utilizzo degli ultimi modelli presentati dipende dalle proprietà di convessità che possiede il sottoproblema. In aggiunta, come si potrebbe aver notato, la particolare costruzione del modello fa sì che i vincoli imposti non rendano necessario aggiungere la funzione obiettivo. I modelli fanno uso della proprietà mostrata nella Proposizione 2.1.2, che unita ai vincoli di continuità tra tratti rettilinei e archi di circonferenza permette di individuare univocamente il cammino ottimo.

Proprietà di convessità del problema

Richiamiamo brevemente i risultati ottenuti in [12], dove viene mostrata la convessità locale della funzione lunghezza del cammino di Dubins che passa per una sequenza di punti. Il lavoro [12] e di conseguenza le considerazioni che ne trarremo prendono in analisi la casistica in cui i punti consecutivi si trovano a distanza maggiore di $4r$ e di conseguenza i cammini ottimi tra ogni due punti sono del tipo CSC. Richiamiamo dal paragrafo precedente la funzione distanza d e riscriviamo $d(\theta_0, ..., \theta_N) = d_0^1(\theta_0, \theta_1) + ... + d_{N-1}^N(\theta_{N-1}, \theta_N)$. Ora definiamo $d_T(\theta_0, \theta_1)$ come la funzione lunghezza del tratto $T \in \{LSR, RSL, LSL, RSR\}$ che congiunge i punti (u_0, θ_0) e (u_1, θ_1). Ogni funzione d_T è due volte differenziabile in (θ_0, θ_1) pertanto è possibile calcolare le derivate parziali e l'Hessiana di d_T che assumono i seguenti valori:

- $\dfrac{\partial d_T(\theta_0, \theta_1)}{\partial \theta_i} = \mu_{i=0}\mu_{C_i=R}(1 - cos(\alpha_i))$
- $\dfrac{\partial^2 d_T}{\partial \theta_i \partial \theta_j} = \delta_{i,j} sin(\alpha_i) + \dfrac{sin(\alpha_i)sin(\alpha_j)}{s}$
- $det(H(d_T)) = sin(\alpha_0)sin(\alpha_1)\left(1 + \dfrac{sin(\alpha_0) + sin(\alpha_1)}{s}\right)$

dove α_i denota la lunghezza dell'i-esimo arco di circonferenza e s quella del tratto rettilineo, $\mu_A = 1$ se la condizione A è vera, -1 altrimenti. Per ulteriori chiarimenti si veda [12].

Proposizione 2.4.1. *Se entrambi gli archi del cammino CSC da (u_0, θ_0) a (u_1, θ_1) hanno lunghezza totale strettamente minore di π allora tutti gli altri cammini CSC differenti hanno lunghezza strettamente maggiore.*

Dimostrazione. ($Cenni$) Consideriamo due cammini distinti T e $T' \in \{LSR, RSL, LSL, RSR\}$ che collegano (u_0, θ_0) e (u_1, θ_1) in maniera tale che gli archi di T abbiano lunghezza strettamente minore di π. Quello che si riesce a mostrare per via geometrica è che il tratto rettilineo di T sarà più corto di quello di T' e lo stesso vale per la lunghezza totale dei tratti curvi da cui segue la tesi. □

Ora, per $\alpha \in (0, \pi]$ denotiamo con $L_i^{i+1}(\alpha)$ l'insieme degli angoli $(\theta_i, \theta_{i+1}) \in (\mathbb{S}^1)^2$ con $\mathbb{S}^1 = \mathbb{R}/2\pi\mathbb{Z}$ tali per cui entrambi gli archi di circonferenza del cammino ottimo tra (u_i, θ_i) e (u_{i+1}, θ_{i+1}) abbiano lunghezza strettamente minore di α. L'insieme (aperto) è ben definito e la Proposizione 2.4.1 ne garantisce l'unicità.

Proposizione 2.4.2. *Se i punti consecutivi di una sequenza distano tra loro più di* $4r$ *la funzione lunghezza* d_i^{i+1} *è* C^2 *e localmente strettamente convessa su* $L_i^{i+1}(\pi)$

Dimostrazione. Senza perdere di generalità consideriamo d_0^1. La distanza maggiore di $4r$ unita al fatto che $(\theta_0, \theta_1) \in L_1^2(\pi)$ ci permettono di dire che esiste unico il cammino T ottimo del tipo CSC tra (u_0, θ_0) e (u_1, θ_1). Supponiamo dapprima che entrambi gli archi di circonferenza abbiano misura non nulla. Osserviamo che al variare di θ_0 e θ_1 la lunghezza degli archi di circonferenza cambia in maniera continua purché rimanga in $(0, 2\pi)$. Quindi in un intorno di (θ_0, θ_1) la lunghezza degli archi di circonferenza del cammino T rimane in $(0, \pi)$ quindi per la Proposizione 2.4.1 d_1^2 e d_T coincidono localmente. Quindi, poiché α_0 e α_1 le lunghezze degli archi di circonferenza stanno in $(0, \pi)$, dalle equazioni ricavate in precedenza $\dfrac{\partial^2 d_T}{\partial \theta_0^2}$ e $det(H(d_T))$ sono positivi. Pertanto d_T è definita positiva e d_T e d_0^1 localmente strettamente convessi in (θ_0, θ_1). Si veda [12] per i casi in cui $\alpha_0 \alpha_1 = 0$. □

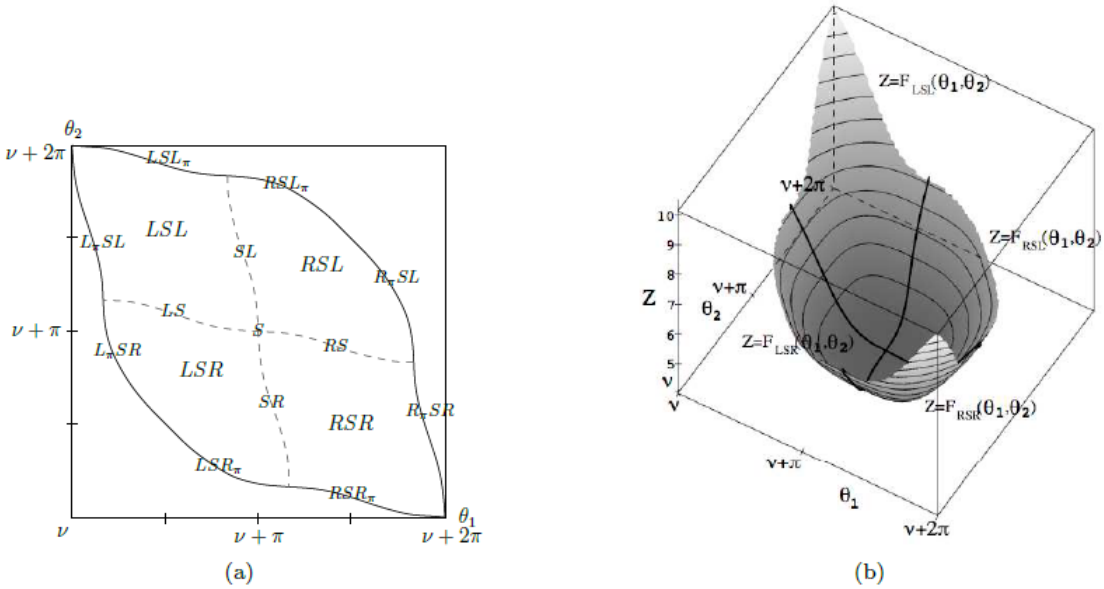


Figura 2.3: (a) $L_1^2(\pi)$ per $|u_1 u_2| = 4$ dove ν è l'angolo di $\overrightarrow{u_2 u_1}$, (b) Grafico tridimensionale di $F = d_1^2$ nello stesso dominio

Denotiamo ora con $\mathcal{L}(\alpha) \subset (\mathbb{S}^1)^{N+1}$ gli insiemi di $(\theta_0, ..., \theta_N)$ tali per cui il cammino ottimo tra $(u_0, \theta_0), ..., (u_N, \theta_N)$ ha tutti gli archi di circonferenza tra punti consecutivi di lunghezza minore di α. Il cammino ottimo per la sequenza di punti è la concatenazione dei cammini ottimi tra (u_i, θ_i) e (u_{i+1}, θ_{i+1}) per $i = 0, ..., N$.

Teorema 2.4.1. *Se i punti consecutivi di una sequenza distano tra loro più di* $4r$ *la funzione lunghezza* $d(\theta_0, ..., \theta_N)$ *è* C^2 *e localmente strettamente convessa su* $\mathcal{L}(\pi)$

Dimostrazione. Si veda [12] □

Il sottoproblema risulta quindi convesso ma richiede un'adeguata inizializzazione della quale discuteremo più avanti. Nonostante le proprietà appena siano di certo un grosso vantaggio c'è da fare i conti con delle difficoltà che emergono non appena si tenta di risolvere concretamente i problemi.

Inammissibilità e cono inammissibile

Nonostante i vantaggi presentati c'è da fare i conti con una problematica. Può succedere che determinate configurazioni non siano adatte per congiungere una sequenza di punti, infatti, dopo aver raggiunto u_i con $i \in \{1, ..., N-1\}$ in un determinato modo, raggiungere u_{i+1} potrebbe risultare impossibile rendendo l'istanza inammissibile. Per evitare che ciò si verifichi proponiamo una nuova possibilità di affrontare il problema. Come abbiamo già osservato, ogni punto u_i con $i \in \{1, ..., N-1\}$ può essere raggiunto dopo aver percorso un ultimo tratto di circonferenza più o meno lungo. Scegliendo come funzione obiettivo la minimizzazione o la massimizzazione di $\sigma_i\beta_i$ possiamo individuare la lunghezza minima e massima dell'ultimo tratto di circonferenza. Chiaramente sia il valore minimo $\beta_i(min)$ che quello massimo $\beta_i(max)$ sono compresi tra 0 e π e in buona parte dei casi si osserva che $\beta_i(min) = 0$ e $\beta_i(max) = \pi$. Tutti i possibili valori di β_i oscillano tra i bound $\beta_i(min)$ e $\beta_i(max)$, e al variare di essi è possibile individuare una sorta di area cieca in cui non può essere collocato u_{i+1} poiché non può essere raggiunto dalla configurazione in questione. Infatti, una volta raggiunto il punto u_i con un angolo β_i, si proseguirà il cammino con un ulteriore arco di circonferenza di lunghezza β_i prima del seguente tratto rettilineo. A partire da queste informazioni siamo in grado di capire se il punto successivo può essere raggiunto in maniera ottimale dal nostro cammino oppure no. In caso negativo è necessario eliminare la sotto-politica così da evitare l'inammissibilità al punto successivo. Siamo così in grado di generare, data una qualunque sotto-politica, un "cono inammissibile" (si veda la Figura 2.4) in cui appunto non può essere collocato il punto u_{i+1}.

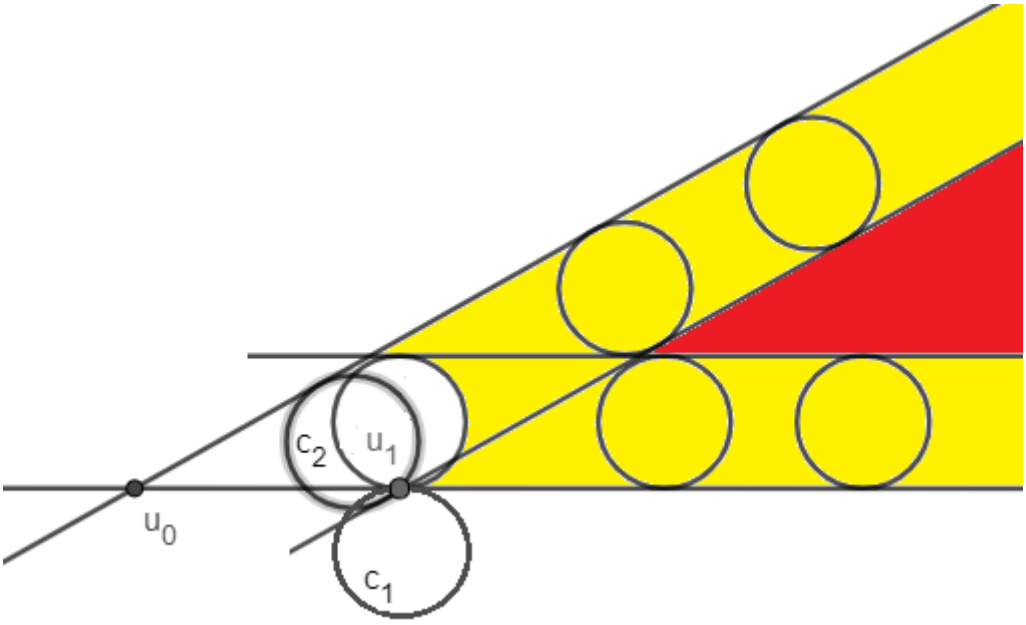


Figura 2.4: In rosso il cono inammissibile di u_1 (curva R nel punto) in cui non può trovarsi u_2. In giallo è indicata la porzione di spazio in cui è necessario vincolare il tipo di curva in u_2. Nell'immagine $\beta_1(min) = 0$ che quindi rappresenta la lunghezza percorsa sulla circonferenza c_1 mentre $\beta_1(max) = \pi$, in questo caso la circonferenza c_2 viene percorsa per intero per tutta la sua lunghezza 2π.

C'è poi un'ulteriore accortezza da considerare, infatti se il punto u_{i+1} finisce nella regione prossima al "cono inammissibile" (cioè se u_{i+1} si trova nella parte colorata in giallo nella Figura 2.4) dobbiamo obbligatoriamente vincolare anche il tipo di curva nel punto u_{i+1}. Solo in questo modo, considerando queste caratteristiche riusciamo ad evitare i casi di inammissibilità.

Metodi risolutivi

Come prima cosa, prima di risolvere il problema dobbiamo capire in che modo verificare l'inammissibilità. Per farlo apportiamo alcune modifiche al quinto modello PNL presentato ottenendo il seguente **modello per la ricerca dell'ammissibilità:**

$$
\begin{cases}
min \quad r & \\
s.t. \quad x_i = x_c^i + cos(\gamma_i) & i = 1, ..., j-1 \\
\qquad y_i = y_c^i + sin(\gamma_i) & i = 1, ..., j-1 \\
\qquad x_u^i = x_c^i + cos(\gamma_i - \beta_i) & i = 1, ..., j-1 \\
\qquad y_u^i = y_c^i + sin(\gamma_i - \beta_i) & i = 1, ..., j-1 \\
\qquad x_v^i = x_c^i + cos(\gamma_i + \beta_i) & i = 1, ..., j-1 \\
\qquad y_v^i = y_c^i + sin(\gamma_i + \beta_i) & i = 1, ..., j-1 \\
\qquad x_j = x_c^j + r \cdot cos(\gamma_j) & \\
\qquad y_j = y_c^j + r \cdot sin(\gamma_j) & \\
\qquad x_u^j = x_c^j + r \cdot cos(\gamma_j - \beta_j) & \\
\qquad y_u^j = y_c^j + r \cdot sin(\gamma_j - \beta_j) & \\
\qquad \gamma_i + \sigma_i * pi/2 = \theta_i + \beta_i & i = 1, ..., j \\
\qquad \theta_{i+1} = \theta_i + 2 * \beta_i & i = 1, ..., j-1 \\
\qquad x_u^1 = x_0 + l_1 * cos(\theta_1) & \\
\qquad y_u^1 = y_0 + l_1 * sin(\theta_1) & \\
\qquad x_u^i = x_v^{i-1} + l_i * cos(\theta_i) & i = 2, ..., j \\
\qquad y_u^i = y_v^{i-1} + l_i * sin(\theta_i) & i = 2, ..., j
\end{cases}
$$

In questo caso viene lasciata la possibilità di raggiungere l'ultimo punti u_j tramite un arco di circonferenza di raggio $r \geq 1$. Minimizzando r siamo in grado di capire se il punto può essere raggiunto o meno dallo specifico cammino. Quando otteniamo $r = 1$ significa che u_j può essere raggiunto e si può estendere il rispettivo cammino che terminava in u_{j-1} fino appunto ad u_j. Quando invece $r > 1$, significa che u_{j+1} non può essere raggiunto tramite la configurazione in questione. Sarà necessario risolvere il modello considerando entrambe le possibilità di arrivo (R o L) nell'ultimo punto u_j perché può darsi che il punto si trovi nella parte di cono inammissibile evidenziata in giallo nella Figura 2.4 e che quindi sia raggiungibile solamente imponendo un determinato tipo di direzione. In questo frangente le soluzioni individuate dalla risoluzione del modello sarebbero in un caso $r = 1$ e nell'altro $r > 1$. Quando $r > 1$ si scarta la politica mentre quando $r = 1$ è possibile proseguire. Sappiamo inoltre che se $r = 1$ sia quando l'ultimo punto viene raggiunto con una curva a destra che quando viene raggiunto con una curva a sinistra, allora significa che l'ultimo punto può essere raggiunto tramite un tratto rettilineo senza percorrere nessun arco di circonferenza.

Considerazioni geometriche

Facciamo qualche considerazione in merito alla disposizione geometrica dei punti col fine di stabilire qualche regola per prevedere il tipo di cammino tra due punti consecutivi effettuando una sorta di precompilazione che alleggerisca la complessità dei problemi non lineari. Quando i punti sono "vicini" tra loro non siamo in grado di fornire una regola generale per intuire il percorso ottimo, infatti un tipo di percorso che potrebbe sembrare ottimale per congiungere alcuni punti potrebbe poi rivelarsi del tutto inadatto per il proseguimento del cammino. A tal proposito è emblematico il Problema 3 in Appendice A dove si può osservare come l'ultimo tratto sia composto da un percorso in un certo senso controintuitivo. Osservando la geometria degli ultimi tre punti ci si potrebbe aspettare che il cammino ottimo compia un tratto più diretto per congiungerli.
Cerchiamo invece di capire cosa succede quando i punti si trovano abbastanza "distanti" e quindi quando tra due punti consecutivi c'è una distanza euclidea superiore a quattro volte il raggio di curvatura. Supponendo che valga questa proprietà, grazie alla Proposizione 2.1.1 sappiamo che ogni tratto è del tipo RSR, RSL, LSL o LSR; cerchiamo allora di intuire se è possibile, analizzando un po' più approfonditamente il problema, ricavare informazioni più precise senza risolvere il modello matematico completo. L'idea di base sta nel cercare di capire con che orientazione viene percorso ogni punto (R o L) osservando la posizione del punto precedente e di quello successivo. Siano quindi u_1, u_2, u_3 tre punti consecutivi fissati che distano più di $4r$ l'uno dall'altro: stiamo cercando di scoprire se a priori siamo in grado di dire qualcosa sull'orientazione del tratto che passa per il punto u_2. Intuitivamente siamo portati a pensare che se ci posizioniamo su u_1 rivolti verso u_3 e in questo modo u_2 si trova sulla nostra sinistra, allora l'orientazione di passaggio al punto u_2 è R mentre nel caso in cui u_2 si trova alla nostra destra l'orientazione è L. Purtroppo, nonostante questa intuizione sia valida per la maggior parte dei cammini, esistono delle casistiche particolari in cui tale proprietà non vale.

Esempio: consideriamo il cammino con raggio di curvatura $r = 1$ che parte da $u_0 = (0, 0)$ con orientazione $\theta_0 = \frac{\pi}{2}$ e passa per $u_1 = (4, 0.5), u_2 = (8, 0), u_3 = (4, -0.5)$ e termina in $u_4 = u_0 = (0, 0)$ con orientazione $\theta_4 = \theta_0 = \frac{\pi}{2}$. Chiaramente tra ogni due punti consecutivi c'è una distanza maggiore di $4r$ e u_1 si trova alla sinistra della retta che passa per u_0 e u_2 (alla sinistra perché facciamo riferimento ad un osservatore posto su u_0 che guarda in direzione u_2). Nel cammino ottimo però l'orientazione di passaggio al punto u_1 è L. Allo stesso modo osserviamo una situazione analoga nel punto u_3. Nelle figure seguenti, che cercano di chiarire tale esempio, sono riportati in nero i segmenti tangenti al punto in cui avviene il cambio tra tipi differenti di tratti (R, L o S).
Mostriamo però che in questo caso la sola posizione dei punti non è una condizione sufficiente per la scelta del tipo di curva in u_1. Infatti, se consideriamo u_0, u_1, u_2 come nel caso precedente, ponendo $\theta_0 = 0$ e poniamo per esempio $u_3 = (10, 0)$ con $\theta_3 = 0$, otteniamo che la curva in u_1 cambia orientazione rispetto al caso precedente. Questo ci mostra che avendo come riferimento solo i punti di passaggio non è facile derivare una regola generica sul tipo di cammino ottimo. Allo stesso modo, al lettore più attento non dovrebbe essere sfuggito il fatto che nell'esempio mostrato la proprietà congetturata fallisce poiché i tre punti consecutivi sono poco scostati dall'essere allineati. Cerchiamo allora una regola ancor più generale, per rendere

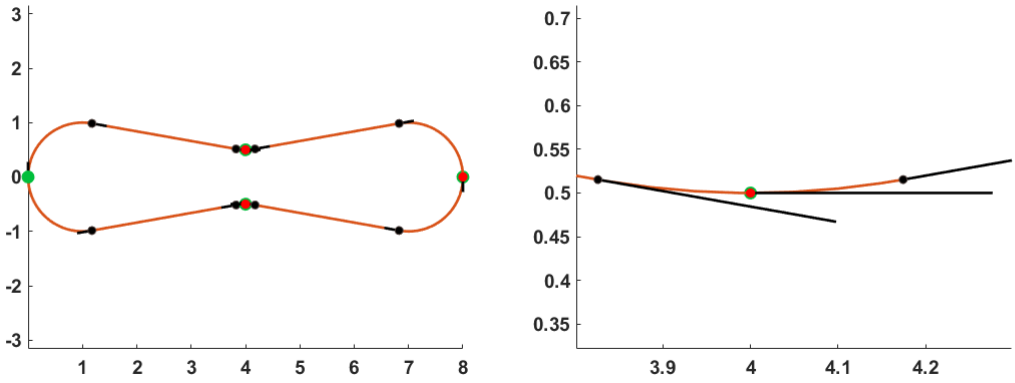


Figura 2.5: A sinistra il cammino ottimo con curva di tipo L in u_1 e u_3 mentre a destra uno zoom sul punto $u_1 = (4, 0.5)$

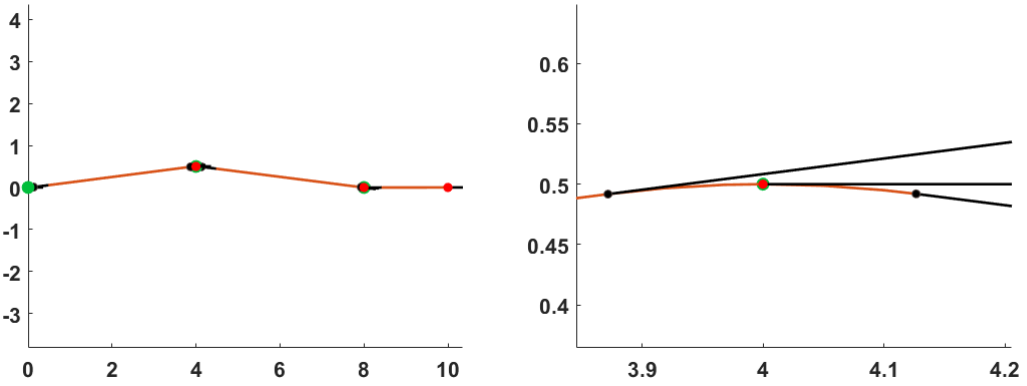


Figura 2.6: A sinistra il cammino ottimo con curva di tipo R in u_1 mentre a destra uno zoom sul punto $u_1 = (4, 0.5)$

possibile, almeno in alcuni casi, una precompilazione. In letteratura, l'unica indicazione proposta è fornita da Goaoc, Kim e Lazard in [12]. Nel loro lavoro, che approfondisce la casistica in cui la distanza tra ogni due punti consecutivi è maggiore di $4r$, viene stabilita la proprietà che stiamo per presentare. Come prima cosa è però necessario enunciare la seguenti definizioni:

Definizione 2.4.1. *Diciamo che un cammino poligonale tra u_{i-1}, u_i e u_{i+1} compie una* ***svolta brusca*** *in u_i se il triangolo $u_{i-1}u_iu_{i+1}$ è acuto in u_i e se la distanza di u_{i-1} dal segmento u_iu_{i+1} o la distanza di u_{i+1} dal segmento $u_{i-1}u_i$ è al più $4r$.*

Definizione 2.4.2. *Dato γ il cammino ottimo che visita ordinatamente le configurazioni (u_0, θ_0), (u_1, θ_1),...,(u_N, θ_N), dico che la* ***classe del cammino ottimo*** *è negativa al punto i (in notazione $\xi_i(\gamma) = -$) se la tangente di γ al punto u_i si trova nel cono formato dai vettori $\overrightarrow{u_iu_{i-1}}$ e $\overrightarrow{u_{i+1}u_i}$. In caso contrario la* ***classe del cammino ottimo*** *è positiva ($\xi_i(\gamma) = +$).*

Proposizione 2.4.3. *Sia γ il cammino ottimo che visita ordinatamente $u_0, u_1, ..., u_N$ e assumiamo che tra ogni due punti consecutivi ci sia una distanza di almeno $4r$, se in u_i* ***non*** *c'è una svolta brusca allora $\xi_i(\gamma) = +$.*

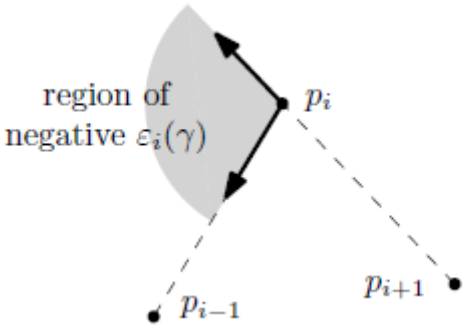


Figura 2.7: Esempio delle regioni delle classi del cammino ottimo

Dimostrazione. Si veda Lemma 7.1 di [12]. □

Per quanto ci permetta di escludere alcune direzioni, questa proprietà non ci permette di asserire con certezza il verso con viene percorsa la curva nel punto u_i. Inoltre, quando l'angolo che forma la poligonale $u_{i-1}u_iu_{i+1}$ in u_i è ottuso, la direzioni che vengono eliminate sono poche e coincidono con le direzioni che verrebbero eliminate mediante uno studio dell'angolo tramite tecniche di discretizzazione. Anche se nella maggior parte dei problemi in cui la distanza tra i punti è superiore a $4r$ risulta evidente con un rapido sguardo il modo in cui vengono percorsi gli archi di circonferenza, non è altrettanto facile ricavare una regola rigorosa ed efficace che ci permetta di ottenere tali informazioni in maniera analitica.

L'inizializzazione delle variabili

Qualunque sia la tecnica adottata, che sia la programmazione dinamica che presenteremo nel prossimo paragrafo oppure la risoluzione "a scatola chiusa" del modello, l'obiettivo finale è quello di identificare univocamente la configurazione ottima e la sua lunghezza. È un fatto noto che i solutori non lineari, su cui stiamo facendo affidamento per la risoluzione delle istanze, necessitino di un'adeguata inizializzazione per ottenere buoni risultati. I vari modelli presentati fino ad ora, se non inizializzati adeguatamente, portano a situazioni di ottimalità locale come già mostrato nel Paragrafo 2.3 o nella peggiore delle ipotesi a casi di inammissibilità, dove va sottolineato che l'inammissibilità in questi casi non è vera e propria, nel senso che la soluzione ottima esiste ma il solutore non riesce ad ottenerla. L'intento è quello di cercare di ridurre al minimo se non annullare queste possibilità, per farlo sarà necessario scegliere un'inizializzazione adatta.

In generale se non vengono forniti dei dati iniziali o vengono forniti solo in parte, **Knitro** si occupa autonomamente di generarli. Il valori che assumono le variabili inizializzate automaticamente sono scelti in maniera casuale ma tali per cui i bound siano soddisfatti. Per quanto lasciare l'inizializzazione automatica delle variabili risulti in alcuni casi comodo ed ugualmente efficace, nei problemi non lineari questo può creare non poche difficoltà di convergenza. Una buona inizializzazione può fare la differenza e permettere al solutore di individuare la soluzione ottima con meno difficoltà. Cominciamo ad escogitare una prima scappatoia per risolvere i problemi di inizializzazione; una possibilità ci viene offerta direttamente dal solutore **Knitro** che tra le varie opzioni concede la possibilità di adottare un approccio multistart.

Come già indicato nei paragrafi precedenti questa tecnica permette in alcuni casi di aggirare le difficoltà. In generale quello che fa il solutore è inizializzare il modello con dei valori iniziali differenti l'uno dall'altro scelti ancora una volta in maniera coerente con i bound sulle variabili. In questo modo, con più punti di partenza aumentano anche le possibilità che almeno uno di essi porti all'ottimo. Tale tecnica come abbiamo già potuto osservare permette di migliorare discretamente le performance dell'algoritmo individuando delle soluzioni ottime che non verrebbero invece raggiunte da un unico lancio del solutore. Capita però delle volte che nemmeno questa tecnica sia sufficiente, in questi casi nonostante le brillanti performance dei solutori è necessario fornire manualmente l'inizializzazione. La tecnica che abbiamo adottato è per la verità molto semplice quanto efficace. Quando dobbiamo risolvere un modello che ha più di tre o quattro punti e non vogliamo che le variabili siano inizializzate in maniera automatica è sufficiente sfruttare i risultati forniti dal passaggio precedente. Ogni nuovo punto da raggiungere incrementa di dieci unità il numero delle nuove variabili pertanto sono sufficienti pochi punti per ottenere dei modelli con una già modesta complessità. In definitiva gli accorgimenti da adottare sono i seguenti: quando si comincia a risolvere il modello con solamente i primi tre o quattro punti è sufficiente lasciare che il solutore inizializzi in maniera automatica le variabili che saranno ancora in numero contenuto. Quando poi andiamo ad aggiungere un nuovo punto al modello è sufficiente che le variabili fino al penultimo punto, ovvero quelle ottenute dall'ottimo del passo precedente, non vengano resettate in modo tale da fungere da inizializzazione per il passo successivo. Per quel che riguarda invece le nuove variabili, che ampliando il modello inevitabilmente si aggiungono a quelle già esistenti le possibilità sono due. Una possibilità è quella di non inizializzarle confidando che la buona inizializzazione delle variabili precedenti sia sufficiente alla convergenza del modello, l'alternativa invece è quella di assegnare almeno alcuni valori per favorire ulteriormente il lavoro del solutore. Elenchiamo quali sono in quest'ultimo caso le operazioni da fare. Innanzitutto, considerando il cammino che arriva in u_i, quando è possibile farlo va fissata con il comando `fix` la variabile $\beta_i = 0$. Chiaramente ciò va fatto nel momento in cui sappiamo valere questa proprietà cioè quando u_i può essere raggiunto sia da destra che da sinistra tramite un arco di circonferenza di raggio unitario. Facciamo notare che il comando `fix` è differente dalla semplice inizializzazione fornita dal comando `let` infatti a differenza di quest'ultimo fissa il valore impedendone la modifica. Solo questo banale accorgimento può rivelarsi sufficiente per il successo nella ricerca dell'ottimo, ma se non dovesse bastare può rivelarsi utile stimare i valori di l_i e θ_i. Per farlo basta inizializzare l_i con la distanza euclidea tra u_{i-1} e u_i mentre θ_i con la direzione del medesimo segmento $u_{i-1}u_i$. Nei test fatti al calcolatore queste tecniche si sono rilevate efficaci; ricordiamo però che come più volte segnalato i modelli proposti sono adatti alla risoluzione di istanze di medie dimensioni e che quindi al crescere di N potrebbero crescere anche le difficoltà di convergenza dei modelli. Quest'ultima frase può suonare un po' strana poiché di fatto ad ogni nuova iterazione, anche con N grande, viene aggiunto solamente un nuovo punto da raggiungere. Tuttavia, più il modello diventa grande e complesso, più l'inizializzazione proposta risulta meno efficace poiché aggiungendo un nuovo punto il cammino totale si scosta da quello inizializzato e di conseguenza la somma degli errori e quindi degli scostamenti dall'ottimo risulta sempre maggiore inficiando in un certo senso sull'efficacia dell'inizializzazione.

2.5 Programmazione dinamica

Supponiamo di trovarci ancora una volta nel caso in cui i punti successivi distano più di $4r$ l'uno dall'altro; se fissiamo la direzione di partenza o quella di arrivo e facciamo variare l'altra, la lunghezza del cammino di Dubins ottimo è una funzione continua su $[0, 2\pi]$. Senza perdere di generalità fissiamo la direzione di partenza e osserviamo cosa succede al variare di quella di arrivo. Se consideriamo una singola tipologia di cammino (RSL, LSR, RSR, LSL), la lunghezza risulta essere continua a tratti. Le discontinuità della lunghezza si compensano vicendevolmente e indicando con RSL*, LSR*, RSR* e LSL* le lunghezze ottime dei rispettivi tratti, la funzione min(RSL*, LSR*, RSR*, LSL*) risulta essere continua nell'intervallo [0,2π]. Fissando un angolo di partenza è possibile, campionando adeguatamente, costruire una partizione dell'angolo di arrivo in cui è nota la lunghezza del tratto e il tipo di curve da compiere per giungere al punto di arrivo in maniera ottimale. Le partizioni così costruite sono formate solitamente da due o tre differenti elementi, è possibile infatti individuare due o tre angoli limite che fungono da spartiacque tra due differenti tipologie di cammino. Non è però facile estendere questo ragionamento al caso in cui entrambe le direzioni siano libere, di conseguenza per individuare il verso di percorrenza delle curve faremo affidamento alla programmazione dinamica.

2.5.1 Costruzione dell'algoritmo

Vediamo ora come costruire l'algoritmo di programmazione dinamica il cui obiettivo è quello di determinare il verso di percorrenza delle curve nel caso in cui i punti distino tra loro una distanza maggiore di $4r$. Enumerando (implicitamente) le scelte discrete, cioè il tipo di curve da percorrere in ogni punto del cammino, grazie alla convessità del sottoproblema continuo esiste ed è unica la soluzione ottima che verrà calcolata grazie ad un solutore PNL. Considerando la sequenza di punti da percorrere, l'obiettivo è appunto quello di individuare i valori $\sigma_i \in \{-1, 1\}$ (rispettivamente R o L) per ogni punto u_i. Si considerino le direzioni di partenza e arrivo libere, in questo modo il cammino ottimo inizia e finisce con un tratto rettilineo permettendoci di non considerare gli archi iniziale e finale e il loro orientamento. L'algoritmo proposto, con i dovuti adattamenti, potrà essere utile anche nel caso in cui la sequenza di punti non è prefissata come nel DTSP (Dubins Travelling Salesman Problem) anche se in tal caso sarà più complicato contenere l'esplosione combinatoria del numero di stati non dominati.
Cerchiamo ora di capire come è costruito uno stato ed identifichiamo quali sono le informazioni necessarie per completare il cammino. L'etichetta di ogni stato è definita da una coppia $\{i, \sigma_i\}$ dove:

- i è l'indice dell'ultimo punto sorvolato u_i;
- $\sigma_i \in \{-1, 1\}$ direzione di sorvolo del punto u_i.

Ad ogni stato è inoltre associato il vettore σ che contiene le direzioni $\{-1, 1\}$ di sorvolo fino al punto i indice dello stato. Per individuare il costo di un percorso associato ad un determinato stato si risolve, quando possibile, il modello PNL in cui si impongono le condizioni $\beta_i = 0, \quad x_u^i = x_i, \quad y_u^i = y_i$, cioè l'ultimo punto u_i viene raggiunto direttamente senza percorrere archi di circonferenza su di esso. Quando non è possibile ottenere il percorso in questo modo è possibile rilassare

le ultime condizioni imposte e permettere al cammino di raggiungere il punto u_i anche tramite un arco di circonferenza. Nel primo caso, ovvero quando il punto può essere raggiunto senza percorrere archi di circonferenza su di esso, il valore di σ_i può variare indipendentemente tra i due valori ammissibili 1 e -1 che generano due stati differenti e al momento equivalenti. L'estensione ad uno stato successivo si genera collegando il punto u_i al punto successivo u_{i+1} con le possibili direzioni di sorvolo di u_{i+1}. Dato quindi un generico stato $\mathcal{L} = \{i, \sigma_i\}$ si genera uno stato $\mathcal{L}' = \{i+1, \sigma_{i+1}\}$ dove il vettore σ è ottenuto dal precedente assegnando il valore prescelto di σ_{i+1}.

NOTA: Quando $u_{i+1} = N$ è necessario imporre le condizioni al bordo finali $\beta_N = 0, \quad x_u^i = x_N, \quad y_u^i = y_N$.

Osservazione: Potrebbe accadere che alcuni valori di σ_i siano noti a priori. In tali casi le rispettive fasi della programmazione dinamica ne risulteranno alleggerite.

Per motivi di efficienza, si vuole che sopravvivano solamente gli stati non dominati. Consideriamo un punto u_i raggiunto mediante due percorsi differenti corrispondenti a due stati $\mathcal{L}' = \{i, \sigma_i\}$ e $\mathcal{L}'' = \{i, \sigma_i'\}$, se indichiamo con $c(\mathcal{L})$ la lunghezza del cammino riferita al generico stato $\mathcal{L}$, se $c(\mathcal{L}') < c(\mathcal{L}'')$ si può mantenere quello di costo minimo $\mathcal{L}'$ solamente se sono verificate le seguenti condizioni:

- qualsiasi soluzione ottenibile da $\mathcal{L}''$ si può ottenere anche da $\mathcal{L}'$;
- il costo di ogni soluzione ottenibile da $\mathcal{L}''$ completandolo con una sotto-politica $\mathcal{L}$ è sempre maggiore o uguale al costo della soluzione ottenibile da $\mathcal{L}'$ completandolo con la stessa sotto-politica $\mathcal{L}$.

Inoltre, per effettuare il test di dominanza tra coppie di stati bisogna anche valutare di quanto può peggiorare $c(\mathcal{L}')$ rispetto a $c(\mathcal{L}'')$ quando i percorsi corrispondenti agli stati $\mathcal{L}'$ e $\mathcal{L}''$ vengono completati con una data sotto-politica $\mathcal{L}$ che porta da u_i alla destinazione u_N. Bisogna innanzitutto distinguere due casistiche: quella più frequente in cui le sotto-politiche $\mathcal{L}'$ e $\mathcal{L}''$ si concludono con un tratto rettilineo e quella in cui ciò non accade. Consideriamo la prima situazione e immaginiamo di dover completare $\mathcal{L}'$ e $\mathcal{L}''$ con la sotto-politica $\mathcal{L}$ agganciando i cammini in u_i in modo da formare un cammino di Dubins. Nel caso peggiore $\mathcal{L}''$ e $\mathcal{L}$ sono allineate e formano già un cammino di Dubins, pertanto il cammino totale ha lunghezza pari a $c(\mathcal{L}) + c(\mathcal{L}'')$ senza nessun aumento. Quello che invece succede tra $\mathcal{L}'$ e $\mathcal{L}$, nel caso peggiore, è l'esatto opposto, cioè le due sotto-politiche giungono in u_i dalla stessa direzione e per raccordare i tratti in modo da formare un cammino di Dubins è necessaria una sorta di inversione a U. In tal caso, come si può evincere dalle seguenti rappresentazioni, l'aumento di costo del cammino risulta essere $(\frac{7}{3}\pi - 2 - 2\sqrt{3})r \approx 1.87r$

Il cammino rappresentato nella Figura 2.8 risulta essere di lunghezza $\frac{7}{3}\pi$, è infatti formato da tre archi di circonferenza consecutivi le cui lunghezze misurano rispettivamente $\frac{1}{3}\pi$, $\frac{5}{3}\pi$ e $\frac{1}{3}\pi$. In questo caso $c(\mathcal{L}) + c(\mathcal{L}')$ non è adeguato a rappresentare il costo totale del cammino che in tal caso non rispetterebbe i vincoli

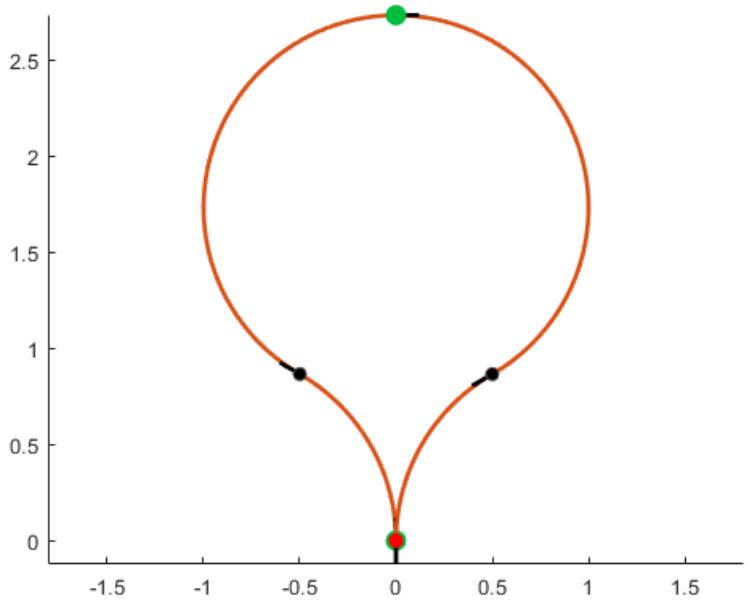


Figura 2.8: Rappresentazione dell'inversione a U nel caso $r = 1$

di anolonomi. L'ultimo tratto (rettilineo) di $\mathcal{L}'$ e il primo (ugualmente rettilineo) di $\mathcal{L}$, entrambi di lunghezza $(1 + \sqrt{3})r$ devono essere rimpiazzati dall'inversione a U della lunghezza di $\frac{7}{3}\pi r$. Di conseguenza la lunghezza totale del cammino risulta essere $c(\mathcal{L}) + c(\mathcal{L}') - 2(1 + \sqrt{3})r + \frac{7}{3}\pi r$.
Questo significa che considerando due stati $\mathcal{L}'$ e $\mathcal{L}''$ con lo stesso indice i tali per cui $c(\mathcal{L}') < c(\mathcal{L}'')$, possiamo eliminare lo stato $c(\mathcal{L}'')$ (dominandolo), se $c(\mathcal{L}'') - c(\mathcal{L}') \geq \frac{7}{3}\pi r - 2(1 + \sqrt{3})r \approx 1.87r$.
Tale bound, per quanto ragionevole, è ulteriormente migliorabile poiché adattabile alle istanze specifiche. A partire dalla direzione di arrivo delle sotto-politiche $\mathcal{L}'$ e $\mathcal{L}''$ al punto u_i possiamo rafforzare il bound che abbiamo visto essere valido nel caso peggiore. Senza sapere la direzione di partenza della sotto-politica $\mathcal{L}$, il caso peggiore sappiamo che può verificarsi solamente quando le direzioni di arrivo di $\mathcal{L}'$ e $\mathcal{L}''$ differiscono esattamente di π. Quando invece le direzioni differiscono di un angolo minore di π il bound può essere reso più stringente. Come si può facilmente osservare nella Figura 2.9 il caso peggiore si verifica quando le due sotto-politiche da raccordare formano un angolo nullo tra esse come abbiamo osservato in Figura 2.8. Il caso ottimo, quando invece i cammini da raccordare differiscono di π, non necessita di peggioramenti. È facile osservare come ci sia una decrescita del bound, prima più rapida e poi sempre più lenta (la derivata della funzione in Figura 2.9 è sempre crescente). Date due sotto-politiche $\mathcal{L}'$ e $\mathcal{L}''$ che giungono ad un medesimo punto con direzioni di arrivo che differiscono di un angolo δ, senza sapere nulla sulle direzioni della sotto-politica $\mathcal{L}$ necessaria a completarle possiamo generare un bound migliore di quello trovato in precedenza. Vista la particolare decrescita della lunghezza del raccordo anolonomo in Figura 2.9, sempre considerando $c(\mathcal{L}') < c(\mathcal{L}'')$, possiamo supporre che nel caso peggiore le direzioni del

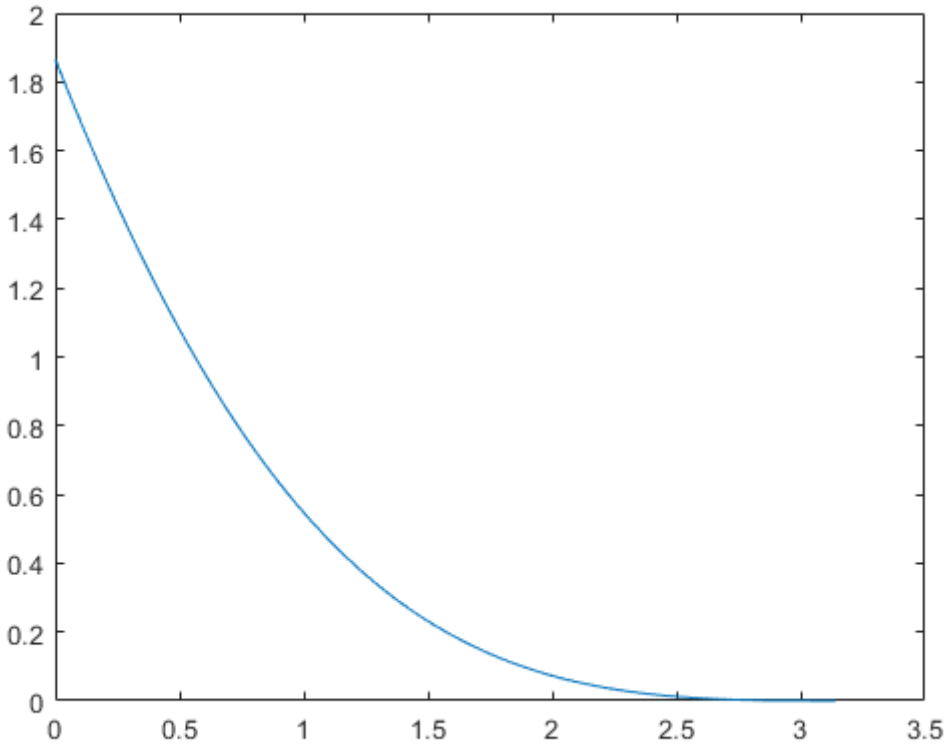


Figura 2.9: Sull'asse delle x si osserva l'ampiezza dell'angolo di cui differiscono la direzione di arrivo di una sotto-politica $\mathcal{L}'$ e quella di partenza della sotto-politica $\mathcal{L}$ necessaria a completare il cammino. Sull'asse delle y si può osservare di quanto è necessario peggiorare il costo del cammino perché soddisfi i vincoli anolonomi.

raccordo tra $\mathcal{L}'$ e $\mathcal{L}$ differiscano di un angolo nullo mentre quelle tra $\mathcal{L}''$ e $\mathcal{L}$ di un angolo δ. In tal caso è sufficiente calcolare, considerando ancora una volta la funzione in Figura 2.9, il valore del peggioramento della lunghezza del raccordo quando l'angolo di differenza in questione è δ. Chiamiamo b_2 l'approssimazione per difetto alla seconda cifra decimale di questo valore. Sapendo che $b_1 = 1.87$ è l'approssimazione per eccesso alla seconda cifra decimale nel caso di una differenza nulla tra gli angoli, il nuovo bound adattato all'istanza del problema risulta essere $b_3 = b_1 - b_2$. Questo significa che nelle ipotesi precedenti possiamo eliminare lo stato $c(\mathcal{L}'')$ (dominandolo), se $c(\mathcal{L}'') - c(\mathcal{L}') \geq b_3 r$.

Consideriamo ora il caso più raro ma di certo più spinoso da trattare in cui il punto u_i viene raggiunto percorrendo un arco di circonferenza su di esso. Quando facciamo il confronto tra due politiche con uno stesso indice i e tra di esse ce n'è una che termina con un tratto curvo non è più valido il bound calcolato nel caso precedente e quindi nel caso peggiore è necessario operare un'inversione a U interamente esterna al cammino precedente. Ciò significa, che nelle ipotesi del caso precedente possiamo eliminare lo stato $c(\mathcal{L}'')$ (dominandolo), se $c(\mathcal{L}'') - c(\mathcal{L}') \geq \frac{7}{3}\pi r$. Il bound non è di certo irresistibile ma siamo consolati dal fatto che questa casistica sia una situazione poco frequente. C'è poi da sottolineare il fatto che risulta poi concreta

la possibilità che non si possa completare il cammino per via della posizione inammissibile di u_{i+1}. (In questi casi il "cono inammissibile" risulta più ampio rispetto al caso medio.)

Di seguito lo pseudocodice relativo all'algoritmo di programmazione dinamica. Con *Modello* facciamo riferimento al quinto modello di PNL proposto nei paragrafi precedenti. Quando l'argomento è semplicemente (i) significa che si considera il modello che termina in u_i con un tratto rettilineo. Quando invece l'argomento è (i, R) o (i, L) significa che il modello termina nel punto u_i con un arco di circonferenza non nullo percorso in direzione rispettivamente destra o sinistra. Con *Ammissibilità* facciamo invece riferimento al modello proposto nel Paragrafo 2.4.4, cioè quello che minimizza il raggio di curvatura dell'ultimo tratto di circonferenza. In questo caso gli argomenti variano tra (i, R) o (i, L), che indicano che il modello termina nel punto u_i con un arco di circonferenza di raggio r rispettivamente a destra e sinistra.
Quando invece si parla di testare la possibile ottimalità, si fa riferimento alle procedure di bounding appena elencate. Come già osservato, quando due differenti configurazioni arrivano al punto u_i con cammini di lunghezza notevolmente diversa, la peggiore può essere scartata. Se questo approccio dovesse mantenere aperte troppe configurazioni non dominate, si potrebbe pensare anche di completare il cammino con una linea spezzata e poi confrontare la lunghezza totale raggiunta con la lunghezza di un cammino ottenuta tramite una configurazione scelta euristicamente.

Algoritmo 1 PROGRAMMAZIONE DINAMICA

DATI: $u_0, ..., u_N$

Risolvi(Modello(1));
Modello(1,R) = Modello(1,L) = Modello(1);
for $i \in 2, .., N-1$ **do**
  while esiste un cammino aperto **c** **do**
    Espandi il cammino aperto **c**
    r_1 = Risolvi(Ammissibilità(i,R));
    r_2 = Risolvi(Ammissibilità(i,L));
    if $r_1 = r_2 = 1$ **then**
      Risolvi(Modello(i));
      Modello(i,R) = Modello(i,L) = Modello(i);
      Testa le possibili ottimalità ed eventualmente **scarta** i cammini
    else if $r_1 = 1$ e $r_2 > 1$ **then**
      Scarta il cammino con la sequenza attuale + (i, L)
      Risolvi(Modello(i,R));
      Testa la possibile ottimalità ed eventualmente **scarta** il cammino
    else if $r_1 > 1$ e $r_2 = 1$ **then**
      Scarta il cammino con la sequenza attuale + (i, R)
      Risolvi(Modello(i,L));
      Testa la possibile ottimalità ed eventualmente **scarta** il cammino
    else if $r_1 > 1$ e $r_2 > 1$ **then**
      Scarta il cammino con la sequenza attuale
    end if
  end while
end for
while esiste un cammino aperto **c** **do**
  Espandi il cammino aperto **c**
  r_1 = Risolvi(Ammissibilità(N,R));
  r_2 = Risolvi(Ammissibilità(N,L));
  if $r_1 = r_2 = 1$ **then**
    Risolvi(Modello(N));
    Se è il risultato migliore ottenuto fino ad ora **MIN** = Risolvi(Modello(N))
  end if
end while
return MIN

2.5.2 Alcune considerazioni

Possiamo concludere che queste tecniche sono in generale efficaci sulle istanze testate ma tuttavia hanno come prezzo da pagare l'affidamento a solutori non lineari che non sempre performano a dovere. Viste le difficoltà imputabili al fatto di lavorare su un set continuo di angoli e quindi la necessità di dover ricorrere a solutori non lineari, come spesso accade nell'analisi numerica sorge spontanea la possibilità di discretizzare il problema. Nonostante sia stato proposto un algoritmo di programmazione dinamica per la risoluzione del problema, è chiaro che ad ogni passo è necessario risolvere il modello non lineare nella sua totalità. Di conseguenza, a differenza dei classici problemi di cammino minimo su grafo, non possiamo sfruttare interamente la soluzione ricavata al passo precedente inficiando così in un certo senso sull'efficacia della procedura. Per sfruttare questo tipo di ricorsione invece è possibile discretizzare il problema. Nel caso specifico la discretizzazione consiste nel rendere in numero finito gli angoli di passaggio. L'idea è quella di adoperare un campionamento dell'angolo giro in k direzioni ottenendo così al crescere di k un'approssimazione abbastanza fedele del problema continuo.

2.6 Discretizzazioni

Percorrendo la strada della discretizzazione del problema continuo, la prima possibilità da considerare è quella di suddividere ogni angolo in un numero fissato di direzioni, calcolare la lunghezza di tutti i possibili collegamenti ottimi tra le varie direzioni di passaggio ed infine tra tutti i cammini scegliere quello di lunghezza minore. Questo metodo di enumerazione tuttavia risulta, in caso di N grande, computazionalmente proibitivo. Immaginando di discretizzare in maniera uniforme ogni angolo in k direzioni; è evidente come il costo computazionale risulti essere $O(k^{N-1})$. La complessità esponenziale rende impensabile adottare questo metodo nelle situazioni in cui N è grande.
Una prima alternativa può essere quella di adottare algoritmi di cammino minimo su grafi: dopo aver costruito un grafo come quello in Figura 2.10 [4] si individua il cammino minimo utilizzando ad esempio l'algoritmo di ricerca A* [14]. Anche in questo caso però la complessità potrebbe risultare elevata se non proibitiva.

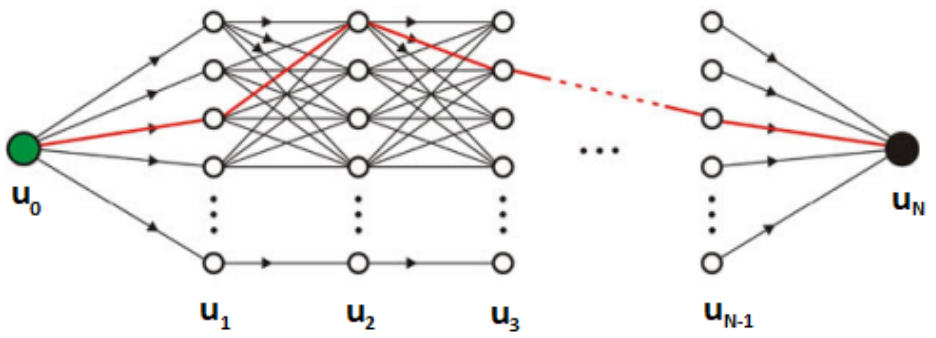


Figura 2.10: Grafo che rappresenta la discretizzazione degli angoli nei vari punti u_i

Osservazione: La complessità esponenziale dipende dalla dimensione N della sequenza di punti ma è anche influenzata dal numero k di discretizzazioni. Vale però la pena osservare che il numero di discretizzazioni k non può assumere valori eccessivamente piccoli. Se vogliamo ottenere delle approssimazioni accurate e non solamente abbozzi del cammino ottimo, k deve assumere un valore intorno a 20 o 30.

C'è poi da tener in considerazione un ulteriore problema che può verificarsi nel tentativo di approssimare la soluzione ottima: in alcune occasioni potrebbe succedere che operando delle discretizzazioni più fitte si ottengano risultati peggiori di discretizzazioni più rade come si può evincere dalla Figura 2.11 [4].

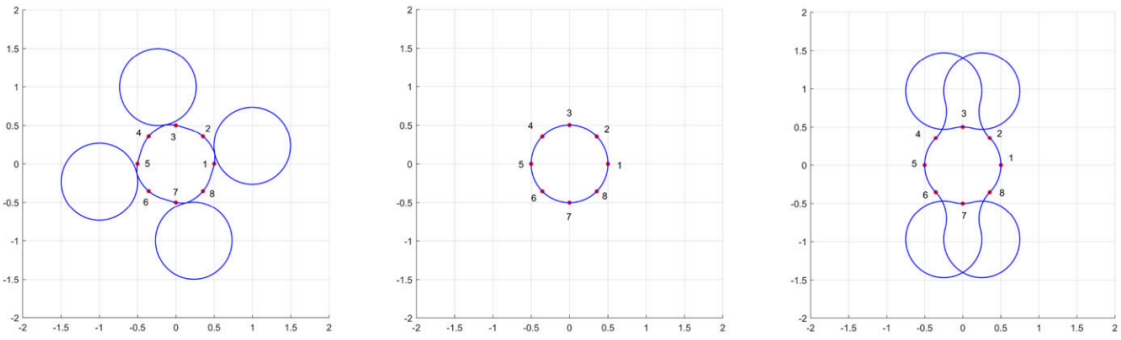


Figura 2.11: Nelle immagini possiamo trovare i cammini ottimi del tour ottenuti rispettivamente con $k = 36$, $k = 72$, $k = 180$.

L'unico modo per aggirare il problema, che si verifica in maniera così evidente solamente quando i punti sono molto vicini tra loro, è quello di aumentare il valore di k poiché chiaramente per $k \longrightarrow \infty$ il problema tende a coincidere con il problema continuo reale.

Nonostante le varie problematiche insorte, non accantoniamo l'idea della discretizzazione, è possibile infatti, sfruttando la natura ricorsiva del problema, costruire un algoritmo in grado di fornirci la stessa soluzione ad un costo computazionale di gran lunga migliore.

2.6.1 Algoritmo di Frego et al.

È l'algoritmo di Frego et al. [10] che ci offre questa possibilità. L'algoritmo, presentato nel 2020, sfrutta la programmazione dinamica per abbattere il costo computazionale complessivo. La naturale ricorsività del problema ci viene in aiuto dandoci la possibilità di dominare una quantità consistente di cammini permettendoci di analizzarne solamente un numero limitato rispetto al totale. Cerchiamo di chiarire il suo funzionamento.
Chiamiamo $L(j, \theta_j)$ la funzione che ci restituisce la lunghezza del sottocammino che parte dal punto j con orientazione θ_j e prosegue fino alla fine. Chiaramente

$$L(j, \theta_j) = \min_{\theta_{j+1},...,\theta_{N-1}} d(\theta_j, \theta_{j+1}) + ... + d(\theta_{N-1}, \theta_N)$$

che può essere riformulato come:

$$L(j, \theta_j) = \min_{\theta_{j+1}} d(\theta_j, \theta_{j+1}) + L(j+1, \theta_{j+1})$$

per $j = N-1, ..., 0$ sapendo che $L(N, \theta_N) = 0$ poiché rappresenta la lunghezza del cammino dall'ultimo punto in poi. In generale, facendo uso di questa ricorsione, ad ogni passo è sufficiente discretizzare due angoli successivi j e $j+1$, e supponendo ancora una volta di discretizzarli in k direzioni ognuno il costo computazionale risulta essere $O(k^2)$ ad ogni passaggio. La complessità totale dell'algoritmo risulta infine essere $O((N-2)k^2)$. La complessità risulta pertanto polinomiale; per evitare che la computazione risulti comunque gravosa potrebbe essere conveniente applicare una prima discretizzazione grossolana al fine di ottenere una soluzione subottimale. Dopo questa prima fase si può procedere ridiscretizzando gli angoli solamente nell'intorno dell'ottimo trovato. Supponendo che nell'ultima discretizzazione effettuata gli angoli differiscono di h l'uno dall'altro, se chiamiamo θ^* la soluzione migliore trovata, nel raffinamento cerchiamo tra gli angoli in $(\theta^* - \frac{3}{2}h, \theta^* + \frac{3}{2}h)$. Questa operazione di raffinamento può essere eseguita più volte al fine di ottenere soluzioni migliori e più vicine all'ottimo esatto del problema. A livello teorico, al crescere del numero di raffinamenti, l'intorno dell'angolo ottimo tende a zero, si presume pertanto di riuscire con un numero adeguato di iterazioni a raggiungere l'ottimo numerico.

NOTA: È chiaro che per utilizzare questa tecnica non è necessario che i punti siano disposti a distanza superiore a $4r$ tra di loro. È altrettanto vero però che più i punti sono vicini più sono necessarie discretizzazioni fitte per non incorrere in errori. Quando i punti sono "distanti" la lunghezza dei cammini varia con continuità cosa che invece non accade quando i punti sono "vicini" e una piccola variazione potrebbe far variare significativamente il risultato.

Risultati numerici

Mostriamo i risultati ottenuti con l'algoritmo di Frego et al. del quale si può trovare lo pseudocodice Matlab utilizzato per produrre i seguenti risultati nell'Appendice B. Si mostrano ora i grafici dei percorsi ottenuti nella risoluzione del Problema 3 dell'Appendice A. Inizialmente si mostrano i cammini ottenuti da un'unica discretizzazione con k = 4, 8, 16, 32 approssimazioni dell'angolo giro.
Successivamente sono mostrati i cammini ottenuti tramite raffinamenti (prima un

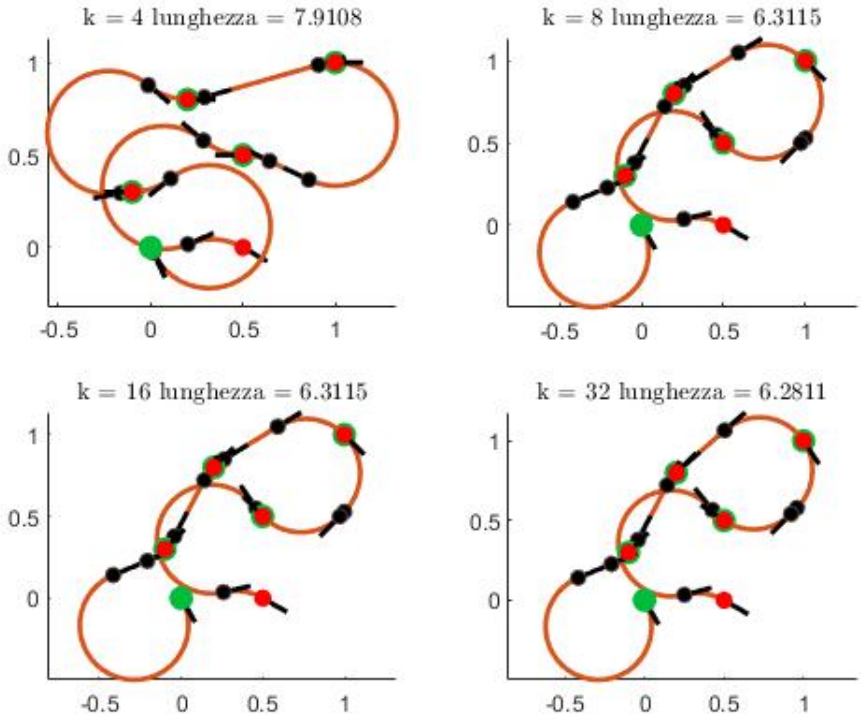


Figura 2.12: Cammini ottenuti mediante un'unica discretizzazione al variare di k

unico raffinamento e poi un raffinamento del raffinamento). In questi casi, in ogni raffinamento si procede con k+1 discretizzazioni dell'intervallo $(\theta^* - \frac{3}{2}h, \theta^* + \frac{3}{2}h)$, dove θ^* è la soluzione ottima calcolata nell'iterazione precedente e h l'ultimo "passo" di discretizzazione. Considerando che nella prima iterazione è stato utilizzato k pari, utilizzando k+1 discretizzazioni nel raffinamento si impone il confronto anche con la precedente soluzione. Questa strategia viene adottata per fare in modo che ad ogni raffinamento la soluzione risulti non peggiore di quella ottenuta al passo precedente. Se nel raffinamento non si facesse questo confronto si potrebbe ottenere una soluzione peggiore rispetto a quella ottenuta al passo precedente, caratteristica certamente poco auspicabile per algoritmi di questo tipo.

Facciamo ora qualche osservazione considerando che il valore della soluzione ottima del Problema 3 è 6.2780. La prima cosa che salta all'occhio è che, escludendo il caso della soluzione ottenuta con un'unica grossolana iterazione con k = 4, il metodo di Frego et al. giunge alle configurazioni migliori della soluzione sin da subito. In un problema come quello affrontato, le prime difficoltà che si riscontrano a livello teorico sono quelle di capire quali possono essere le configurazioni adeguate per ottenere il cammino più breve. Con l'algoritmo di Frego et al. si ha un'efficacia elevata e le configurazioni peggiori vengono subito scartate. Si può osservare inoltre, che in un problema di piccole dimensioni come quello analizzato, i risultati sono accurati e gli errori arrivano facilmente ad essere dell'ordine di 10^{-2} anche senza procedere con discretizzazioni eccessivamente fini. Nei problemi di dimensione non eccessiva come il Problema 3 appena analizzato è interessante, appesantendo il carico computazionale, operare discretizzazioni e raffinamenti molto

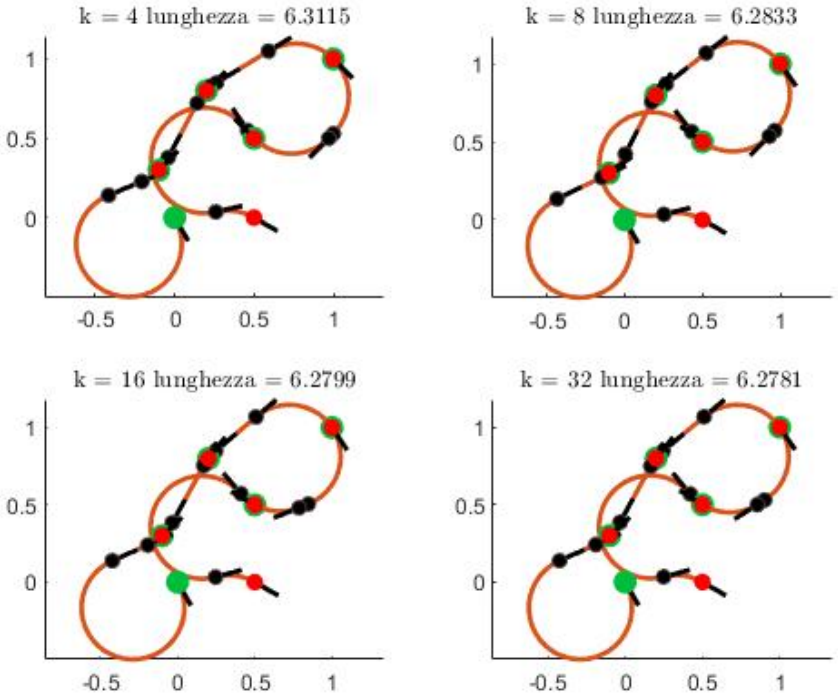


Figura 2.13: Cammini ottenuti mediante un raffinamento al variare di k

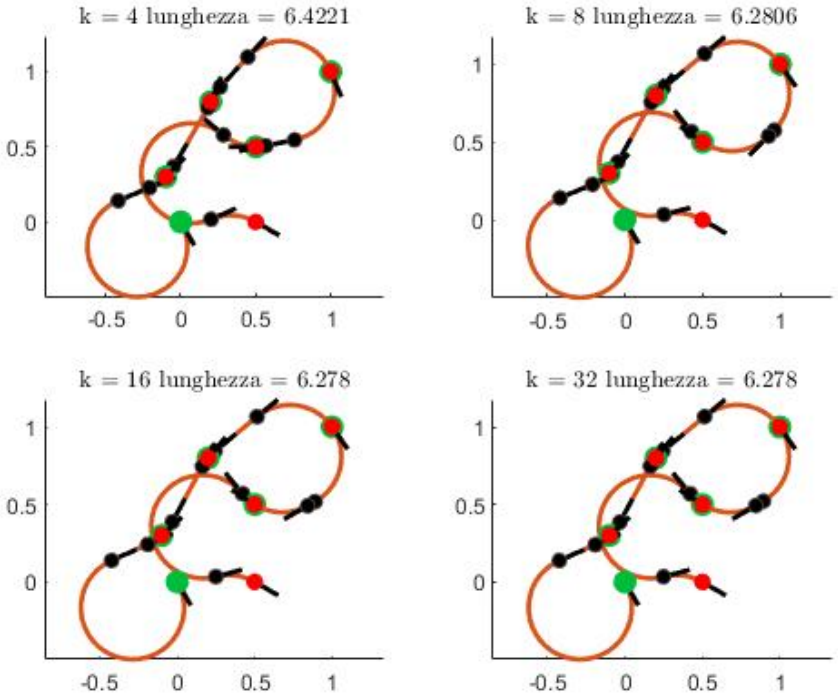


Figura 2.14: Cammini ottenuti mediante un doppio raffinamento al variare di k

fini, con lo scopo di verificare la precisione della soluzione ottenuta rispetto a quella ottima. Consideriamo ora la lunghezza del cammino ottimo approssimato alla dodicesima cifra decimale che risulta essere 6.278034550309, seguendo l'approccio dell'algoritmo Frego et al. con k=100 e due raffinamenti consecutivi si ottiene un cammino della lunghezza di 6.278034551326981, che si discosta dall'ottimo di un errore dell'ordine di grandezza di 10^{-9}. Raddoppiando gli angoli di discretizzazione con k = 200 e due raffinamenti si raggiunge con esattezza anche la dodicesima cifra decimale. In generale l'algoritmo di Frego et al. è stato testato su tutte le istanze dell'Appendice A fornendo ottimi risultati, di seguito riportiamo le soluzioni ottenute nella risoluzione del Problema 4, i risultati ottenuti sulle altre istanze testate non sono riportati per brevità.

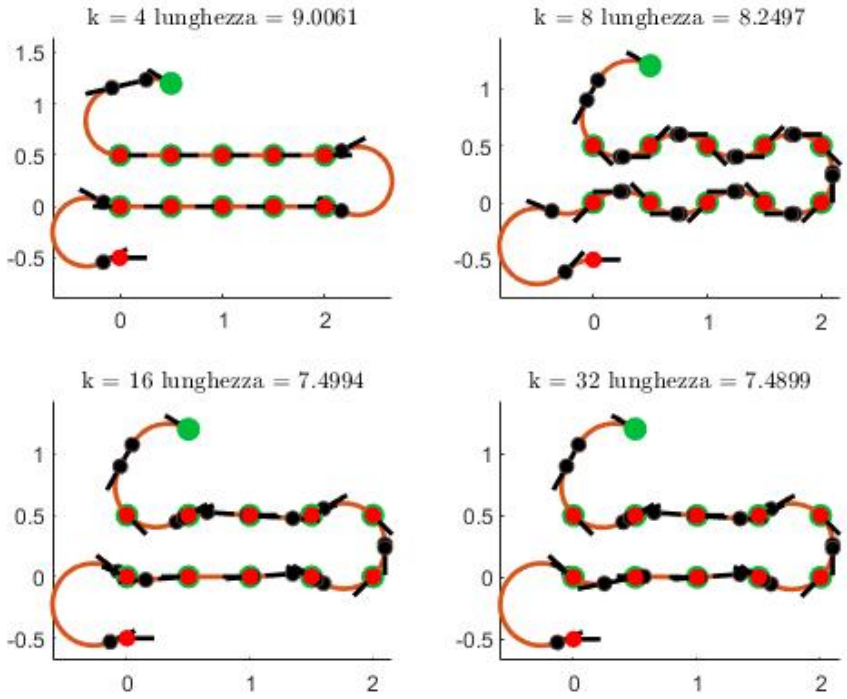


Figura 2.15: Cammini ottenuti mediante un'unica discretizzazione al variare di k

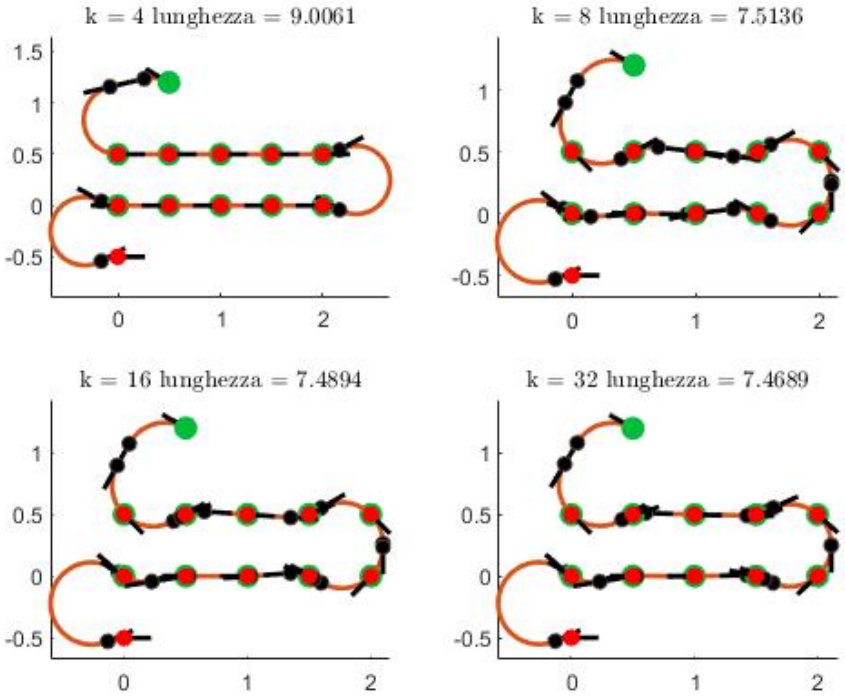


Figura 2.16: Cammini ottenuti mediante un raffinamento al variare di k

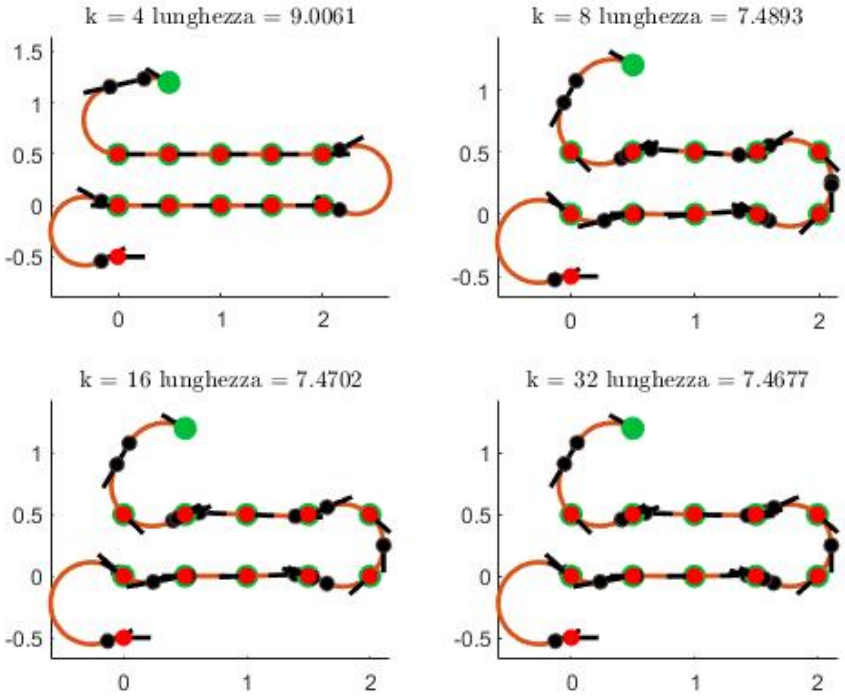


Figura 2.17: Cammini ottenuti mediante un doppio raffinamento al variare di k

2.7 Conclusioni sull'interpolazione di Dubins

Traiamo ora qualche considerazione conclusiva a termine di questo capitolo. In questi paragrafi abbiamo presentato le tecniche risolutive per il problema di interpolazione di Markov-Dubins ovvero la ricerca del cammino di Dubins minimo lungo una sequenza di punti. Abbiamo analizzato differenti approcci al problema di cui nelle prossime righe sintetizzeremo gli esiti. Inizialmente abbiamo affrontato il problema costruendo dei modelli non lineari da dare in pasto a dei solutori non lineari per individuare la soluzione. Un approccio di questo tipo può essere deficitario su alcuni aspetti, primo su tutti la presenza dei già citati ottimi locali. Se in alcuni casi, specialmente i più piccoli, possiamo affrontare in questo modo il problema, generalmente è meglio adottare un'altra strategia. È qui che entra in gioco l'algoritmo di programmazione dinamica da noi proposto (valido quando punti consecutivi distano più di $4r$) che permette di evitare problemi dovuti ad ottimalità locali e permette inoltre di inizializzare adeguatamente il problema mediante lanci consecutivi del solutore come descritto nei paragrafi precedenti. Anche questa tecnica però ha un limite che è dovuto alla complessità computazionale ed al fatto che si debba fare affidamento a solutori che se non inizializzati a dovere potrebbero avere difficoltà di convergenza. L'altra possibilità che abbiamo quando siamo alla ricerca di un'ottima approssimazione ad un costo computazionale ridotto è l'algoritmo di Frego et al. che ci fornisce ottime performance. Discretizzando il problema e sfruttando le caratteristiche ricorsive di tale formulazione riusciamo ad ottenere delle eccellenti approssimazioni anche su problemi di grandi dimensioni. Se non c'è la necessità di ottenere l'ottimo e ci si accontenta di una buona soluzione approssimata l'algoritmo di Frego et al. è probabilmente la miglior tecnica da prendere in considerazione nonché l'unica in grado di risolvere senza difficoltà anche le istanze più grandi. I tempi di calcolo degli algoritmi verranno discussi nel capitolo seguente insieme a quelli delle altre tecniche che verranno presentate.

Capitolo 3

Cammini ottimi per insiemi di punti

Consideriamo ora il problema in cui il veicolo di Dubins deve raggiungere un insieme di punti fissati che devono essere visitati un'unica volta. Quando abbiamo a che fare con un problema di questo tipo, possiamo parlare di DTSP (Dubins Travelling Salesman Problem), una variante del ben più noto TSP (Travelling Salesman Problem) in cui la distanza tra due punti non è quella euclidea bensì quella generata dai veicoli anolonomi di Dubins[1]. In generale possiamo distinguere due diverse varianti: il tour-DTSP e il path-DTSP. Nel primo caso è necessario che il punto di partenza e quello di arrivo coincidano, nel secondo invece tale vincolo viene rilassato. Nel seguito del lavoro ci concentreremo sul tour-DTSP, nel quale partenza e arrivo combaciano e in cui le direzioni di passaggio sono libere. Riassumendo, il problema consiste nel trovare il percorso più breve che passa per tutti i punti fissati, concretamente è necessario individuare la sequenza dei punti e l'orientazione del veicolo durante il passaggio per ognuno di essi; in questo modo il percorso risulta univocamente definito. Si può mostrare che questo problema rientra nella categoria dei problemi NP-difficili [23]. Contrariamente a quanto si possa pensare, il fatto che il DTPS appartenga a tale classe di complessità non può essere banalmente dedotto del fatto che ne faccia parte il TSP. L'aggiunta di un vincolo non sempre rende più complesso il problema, anzi può succedere che lo semplifichi come per esempio accade nel bitonic-TSP [38].

Teorema 3.0.1. *Il DTSP è NP-difficile.*

Dimostrazione. Questo è un corollario della dimostrazione di Papadimitriou sulla NP-completezza del ETSP (Euclidean Travelling Salesman Problem) [25] cioè il problema del commesso viaggiatore in cui la distanza tra due punti nel piano è proprio la distanza euclidea. Papadimitriou è riuscito a costruire una riduzione polinomiale del problema della Copertura Esatta, problema notoriamente NP-completo [17], all'ETSP. In generale data un'istanza del problema della Copertura Esatta, possiamo costruire un'istanza del ETSP e un numero L tali per cui la Copertura Esatta ha soluzione se e solo se l'ETSP ha un tour ottimo di lunghezza minore o uguale a L. Se il problema della Copertura Esatta non dovesse avere una soluzione,

[1] Come già osservato, quella di Dubins non può essere formalmente definita distanza in quanto manca della proprietà di simmetria.

la riduzione di Papadimitriou ci restituisce un'istanza dell'ETSP il cui ottimo ha lunghezza maggiore o uguale a L + δ per qualche $\delta > 0$. Da [31] sappiamo che esiste una costante C tale per cui per ogni istanza $\mathcal{P}$ dell'ETSP con n punti con lunghezza ETSP($\mathcal{P}$) il corrispondente DTSP ha lunghezza ottima minore o uguale a ETSP($\mathcal{P}$) + Cn. Ora, se abbiamo n punti nell'istanza del ETSP costruito nella dimostrazione di Papadimitriou, riscaliamo le distanze di un fattore 2Cn/δ. Pertanto, se il problema della Copertura Esatta ha soluzione, l'ETSP ha ottimo di lunghezza minore o uguale a 2LCn/δ e di conseguenza il corrispondente DTSP avrà lunghezza non superiore a 2LCn/δ + Cn. Se invece la Copertura Esatta non dovesse aver soluzione, la lunghezza ottima per l'ETSP sarebbe maggiore o uguale a 2LCn/δ + 2Cn così come per il DTSP. In conclusione, la tesi è dimostrata riscalando la costruzione di Papadimitriou di un fattore 2LCn/δ e scegliendo 2LCn/δ + Cn al posto di L. □

3.1 Differenti strategie risolutive

Il Teorema 3.0.1 ci permette di osservare, sotto l'ipotesi che $P \neq NP$, che il DTSP non possiede un algoritmo di risoluzione efficiente in tempo polinomiale. È possibile però approcciarsi al problema mediante differenti tecniche in modo da approssimare al meglio la soluzione.

In letteratura sono presenti vari approcci al problema. Principalmente la distinzione tra le classi di algoritmi risolutivi risiede nel fatto di usare o meno la soluzione ottima del relativo ETSP. Dopo aver analizzato una carrellata di algoritmi presenti allo stato dell'arte dei quali discuteremo proprietà e ottimalità delle soluzioni, presenteremo l'estensione dell'algoritmo di programmazione dinamica del capitolo precedente.

3.2 Algoritmi con utilizzo dell'ETSP

Euclidean Travelling Salesman Problem

L'approccio forse più naturale e immediato al problema del DTSP non può che derivare dalla soluzione del problema di cui è una variante. Stiamo chiaramente parlando del TSP ed in particolar modo dell'ETSP, problemi ampiamente studiati in letteratura negli ultimi decenni. Per quanto notoriamente appartengano alla classe dei problemi NP-difficili, la loro rilevanza ne ha garantito un esteso dibattito e di conseguenza un'ampia varietà di tecniche risolutive. Oltre ad una serie di euristiche, esistono algoritmi esatti che garantiscono l'ottimo su istanze ragionevoli con un costo computazionale non elevato. Le tecniche più comuni sono algoritmi di branch-and-bound, algoritmi di programmazione dinamica oppure algoritmi di cutting planes. Senza voler entrare nei dettagli, mostriamo di seguito la formulazione del problema nel linguaggio della programmazione lineare.
Denotiamo la distanza euclidea tra due punti $u_i = \binom{x_i}{y_i}, u_j = \binom{x_j}{y_j}$ come

$$c_{ij} := \| \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \|$$

ed introduciamo le variabili binare

$$x_{ij} = \begin{cases} 1 & \text{se il tratto } u_i u_j \text{ è incluso nel tour} \\ 0 & \text{altrimenti} \end{cases}$$

Denominando con P l'insieme dei punti $u_0, ..., u_N$, possiamo formulare il modello noto come formulazione di Dantzig–Fulkerson–Johnson:

$$\begin{cases} min & \sum_{i \in P} \sum_{j \in P} c_{ij} x_{ij} & \\ s.t. & \sum_{j \in P} x_{ij} = 1 & \forall i \in P \\ & \sum_{i \in P} x_{ij} = 1 & \forall j \in P \\ & \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 & \forall S \subset P \\ & x_{ij} \in \{0, 1\} & \forall i, j \in P \end{cases}$$

Angular Travelling Salesman Problem

Non sempre l'ETSP fornisce un ottimo ordinamento per il DTSP, ciò è dovuto al fatto che la lunghezza di quest'ultimo non dipende solamente dalla distanza euclidea tra i punti. Come abbiamo già osservato nel primo capitolo di questo lavoro, la distanza di Dubins tende ad avere le caratteristiche di quella euclidea solamente per punti distanti tra loro mentre la distanza di Dubins tra punti più vicini ha una dipendenza molto più stretta dalle direzioni di passaggio. Se vogliamo trovare l'ordinamento ottimo nel caso del DTSP può risultare intelligente prendere in considerazione anche la somma delle variazioni di direzione tra punti consecutivi. In un ragionamento come questo verranno predilette le sequenze di punti più allineate delle altre in maniera tale da limitare il più possibile e quindi minimizzare gli archi di circonferenza più lunghi che sono la principale causa dei allungamento del DTSP rispetto all'ETSP.
A questo proposito definiamo il costo rappresentato dal cambiamento di direzione: dati tre punti consecutivi u_i, u_j, u_k e le loro rispettive coordinate $\binom{x_i}{y_i}, \binom{x_j}{y_j}, \binom{x_k}{y_k}$, sia

$$\alpha_{i,j,k} = \alpha(i,j,k) := arcos_{[0,\pi]} \left(\frac{\binom{x_j}{y_j} - \binom{x_i}{y_i}}{\| \binom{x_j}{y_j} - \binom{x_i}{y_i} \|} \cdot \frac{\binom{x_k}{y_k} - \binom{x_j}{y_j}}{\| \binom{x_k}{y_k} - \binom{x_j}{y_j} \|} \right)$$

dove $\cdot$ denota il prodotto scalare.
In aggiunta alle variabili del'ETSP, introduciamo un ulteriore tipo di variabili binarie come segue:

$$y_{ijk} = \begin{cases} 1 & \text{se i tratti } u_i u_j \text{ e } u_j u_k \text{ sono inclusi entrambi nel tour} \\ 0 & \text{altrimenti} \end{cases}$$

A questo punto il modello matematico dell'Angular Travelling Salesman Problem [22] [35] risulta essere il seguente:

$$\begin{cases} min & w_1 \sum_{i\in P}\sum_{j\in P} c_{ij}x_{ij} + w_2 \sum_{i\in P}\sum_{j\in P}\sum_{k\in P} \alpha_{ijk}y_{ijk} \\ s.t. & \sum_{j\in P} x_{ij} = 1 & \forall i \in P \\ & \sum_{i\in P} x_{ij} = 1 & \forall j \in P \\ & y_{ijk} \geq x_{ij} + x_{jk} - 1 & \forall i,j,k \in P \\ & \sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 & \forall S \subset P \\ & x_{ij} \in \{0,1\} & \forall i,j \in P \\ & y_{ijk} \in \{0,1\} & \forall i,j,k \in P \end{cases}$$

Relazioni tra le lunghezze di ETSP e DTSP

Una prima banale osservazione consiste nel fatto che $DTSP(r)_o \geq ETSP_o$ dove $DTSP(r)_o$ e $ETSP_o$ indicano le misure dei cammini ottimi dei rispettivi problemi. Vale inoltre il seguente risultato [31] [32]:

Proposizione 3.2.1. *Dato $r > 0$ esiste un insieme di punti $u_0, ..., u_N$ con $N \geq 1$ tali per cui $DTSP(r)_o \geq ETSP_o + 2\lfloor\frac{N+1}{2}\rfloor\pi r$*

Facciamo ora qualche considerazione sull'upper bound; date due configurazioni qualunque nello spazio ed indicata con D la distanza di Dubins e con d quella euclidea tra esse vale [8] [31]

$$D \leq d + \kappa\pi r \qquad \kappa \in [2.657, 2.658]$$

Se vale inoltre $\eta r \leq d$ cioè la distanza euclidea tra i due punti è superiore a ηr possiamo riscrivere la precedente relazione come

$$D \leq (1 + \frac{\kappa\pi}{\eta})d \qquad \kappa \in [2.657, 2.658]$$

Ora, se un algoritmo per l'individuazione del DTSP fa uso della sequenza ottima dell'ETSP e tra ogni coppia di punti c'è una distanza euclidea superiore a ηr allora vale il seguente upper bound:

$$DTSP(r)_o \leq (1 + \frac{\kappa\pi}{\eta})ETSP_o$$

In questo frangente è interessante osservare che, come avevamo predetto nel primo capitolo, quando la distanza tra due punti diventa molto grande allora la distanza di Dubins tende a quella euclidea come confermato da quest'ultimo risultato.

3.2.1 Alternating Algorithm (AA)

Il primo algoritmo che mostriamo è un'euristica che mediante la sequenza ottima dell'ETSP trova una soluzione ammissibile del DTSP. Tale procedura prende il nome di Alternating Algorithm. In realtà è un processo molto grezzo e banale che tuttavia permette, in casi standard, di ottenere soluzioni del DTSP accettabili. L'algoritmo consiste nei seguenti passi:

- Dato l'insieme di punti $(u_0, ..., u_N)$ si calcola la sequenza ottima $(i_0, ...i_N)$ per l'ETSP che viene assunta come sequenza anche per il DTSP.

- Definisco $\theta_0 :=$ l'orientamento del segmento $i_0 i_1$.

- Per $j \in \{1, ..., N-1\}$, indicando con θ_j la direzione di sorvolo del punto i_j, definisco $\theta_j := \theta_{j-1}$ quando j è dispari, mentre $\theta_j :=$ l'orientamento del segmento $i_j i_{j+1}$ quando j è pari.

- Ora se N è dispari $\theta_N := \theta_{N-1}$, altrimenti se N è pari $\theta_N :=$ l'orientamento del segmento $i_N i_0$.

È evidente quanto questa procedura sia banale, tuttavia ci permette, nel caso in cui $N \geq 3$ di rafforzare il bound precedente con [31]

$$DTSP(r)_{o,AA} \leq (1 + \frac{5\kappa\pi}{6\eta}) ETSP_o$$

3.2.2 Angle Bisector Algorithm (ABA)

Quest'altro algoritmo [8] che si basa nuovamente sulla sequenza di punti fornita dall'ETSP risolve il problema del DTSP-path. Il problema è tuttavia adattabile al caso in cui i punti di partenza ed arrivo non coincidano ed è costituito da tre fasi consecutive:

- Dato l'insieme di punti $(u_0, ..., u_N)$ si calcola la sequenza ottima $(u_0, i_1, ...i_N, u_0)$ per l'ETSP che viene assunta come sequenza anche per il DTSP.

- Ora considerando i primi tre punti consecutivi della sequenza u_0, i_1, i_2 si calcola l'angolo di passaggio θ_1 per il punto intermedio come mostrato in Figura 3.1. Alternativamente denominiamo A, B, C i tre punti consecutivi. Chiamiamo α_{10} l'angolo formato dal raggio BA e α_{21} l'angolo formato dal raggio BC. A questo punto si consideri la seguente formula:
$$\theta_1 = \begin{cases} \frac{(\alpha_{21} + \alpha_{10})}{2} - \frac{\pi}{2} & se \ \ \alpha_{21} < \alpha_{10} \\ \frac{(\alpha_{21} + \alpha_{10})}{2} + \frac{\pi}{2} & altrimenti \end{cases}$$

- Ripetere la seconda fase per ogni tre punti consecutivi nella sequenza ed individuare le rispettive direzioni di passaggio.

- Come ultima operazione calcolare l'angolo in u_0 usando la terna i_N, u_0, i_1.

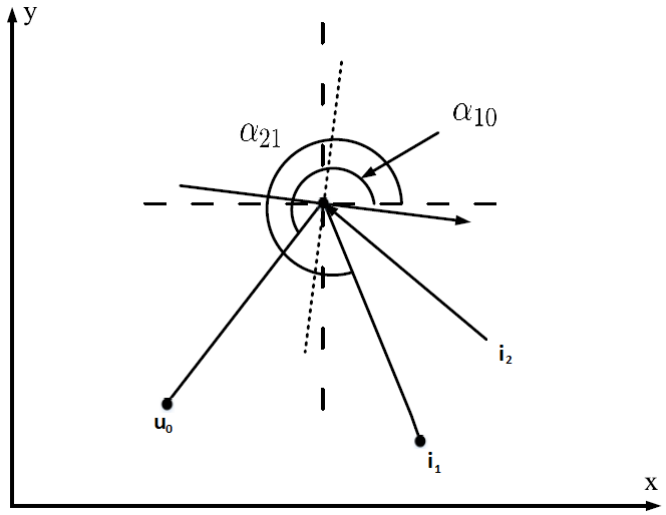


Figura 3.1: Esemplificazione grafica dell'orientamento dell'angolo in i_1

Mostriamo ora che, nel caso in cui ogni due punti consecutivi distino una distanza superiore a $4r$, ABA fornisce una soluzione che nel caso peggiore che differisce dall'ottimo del ETSP per una costante moltiplicativa.

Teorema 3.2.1. *Se la distanza euclidea tra ogni due punti in un fissato insieme è almeno 4r allora la lunghezza del DTSP tour calcolato mediante ABA soddisfa* $DTSP_{o,ABA} \leq 2.57 ETSP_o$.

Dimostrazione. Consideriamo la sequenza di punti fornita dall'ETSP e siano A, B, C, D quattro punti consecutivi nella sequenza. Senza perdere di generalità consideriamo che il segmento che unisce B e C sia orizzontale. In accordo con ABA, i possibili orientamenti degli angoli in B e C risultano appartenere ai quadranti I e IV, a destra rispettivamente delle linee L_1 e L_2 in accordo con la Figura 3.2. Ora concentriamoci sulla lunghezza del cammino di Dubins tra i Punti B e C. Al variare delle possibili direzioni di partenza e arrivo, denotando con a_{ij} le classi di cammini che hanno direzione in B appartenente al quadrante i e direzione in C appartenente al quadrante j, in accordo con [34] possiamo prendere in considerazione solamente due classi differenti. Quello che succede infatti è che tramite trasformazioni ortogonali è possibile mostrare l'equivalenza delle classi a_{11} e a_{44} e delle classi a_{14} e a_{41}. Denotiamo con α e β le direzioni in B e C. Considerando i cammini appartenenti alla classe a_{11}, sappiamo grazie a [34], che i cammini ottimi sono del tipo RSL e quello di lunghezza massima si verifica quando $\alpha = \beta = \frac{\pi}{2}$. In tal caso possiamo indicare il seguente upper bound: $L_{11,RSL} \leq d + 2\pi r$, dove d è la distanza euclidea tra B e C. Poiché nelle ipotesi abbiamo supposto che $d \geq 4r$ possiamo affermare che $L_{11,RSL} \leq (1 + \frac{\pi}{2})d$.
Considerando ora i cammini appartenenti alla classe b_{14}, sappiamo che l'ottimo è un cammino del tipo RSL o RSR. Per quel che riguarda i cammini di tipo RSL, il caso peggiore si verifica quando $\alpha = \frac{\pi}{2}$ e $\beta = 0$ o 2π, mentre per quel che riguarda i cammini di tipo RSR, il caso peggiore si verifica quando $\alpha = \frac{\pi}{2}$ e $\beta = \frac{3}{2}\pi$. In questi casi sono validi i seguenti upper bounds: $L_{14,RSL} \leq d + r + \frac{3}{2}\pi r$ e

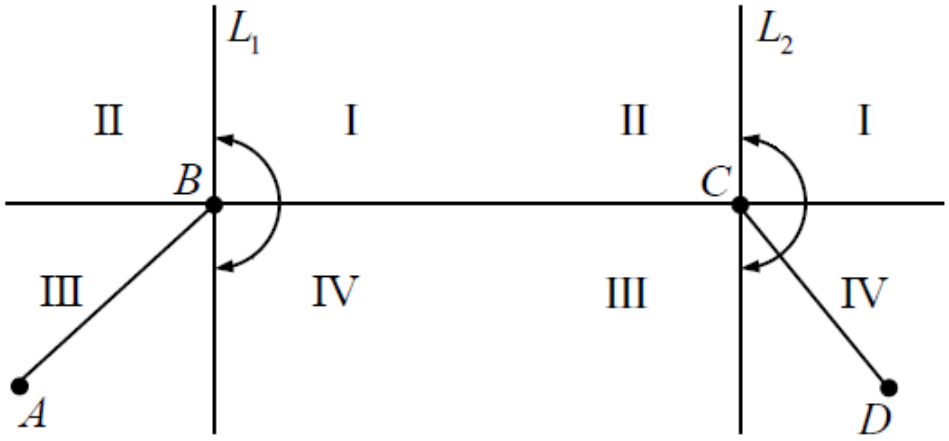


Figura 3.2: Possibili orientamenti degli angoli in B e C

$L_{14,RSR} \leq d+(\pi+2)r$. Considerando poi che $d \geq 4r$ valgono $L_{14,RSL} \leq (1+\frac{1}{4}+\frac{3}{8}\pi)d$ e $L_{14,RSR} \leq (1+\frac{1}{2}+\frac{\pi}{4})d$. Ora, dalle disuguaglianze ottenute possiamo affermare che

$$L_{BC} \leq max\left((1+\frac{\pi}{2})d, (1+\frac{1}{4}+\frac{3}{8}\pi)d, (1+\frac{1}{2}+\frac{\pi}{4})d\right) \approx 2.57d$$

A questo punto, considerando che B e C rappresentano due punti qualunque nella sequenza, tale bound è valido per qualunque coppia di punti consecutivi e da qui si deduce facilmente la tesi. □

Osservazione: Per quanto questo teorema non garantisca un risultato straordinario, ci permette di dominare il bound di AA.

3.2.3 Euristica con algoritmo di Frego et al.

Vista l'efficacia dell'algoritmo di Frego et al. presentato nel capitolo precedente è naturale che venga chiamato in causa anche per la risoluzione di questa variante del problema. La sequenza ottima da dare in pasto all'algoritmo è ancora una volta quella ottenuta mediante la risoluzione all'ottimo dell'ETSP. Scelto un punto di partenza (e quindi di arrivo) abbiamo poi adattato l'algoritmo in maniera tale da rendere libere le direzioni iniziale e finale. Anche in questo caso i risultati sono andati oltre le aspettative come confermano i risultati numerici che ci apprestiamo a presentare.

Risultati numerici

Mettiamo a confronto i risultati ottenuti tramite gli algoritmi presentati. Di seguito le performance e i tempi di calcolo degli algoritmi implementati con Matlab R2021a su un laptop con processore da 2.00 GHz. Per verificare l'efficacia delle tecniche i punti da raggiungere sono stati scelti in maniera casuale all'interno del quadrato $[0,20]\times[0,20]$. Mantenendo costante il raggio di curvatura uguale a 1 abbiamo messo a confronto le varie tecniche in due casi differenti. Nel primo caso non è stato posto nessun vincolo sulla posizione dei punti mentre in un secondo momento abbiamo imposto che la distanza tra punti fosse almeno $4r$. In entrambi i casi all'interno dei grafici è riportata anche la lunghezza del cammino euclideo ottimo che funge da lower bound. I risultati che si possono osservare dai grafici sono frutto di una media tra i risultati di 50 differenti campionamenti casuali pertanto possono essere considerati esemplificativi dell'andamento generale degli algoritmi e non specifici di una particolare istanza.

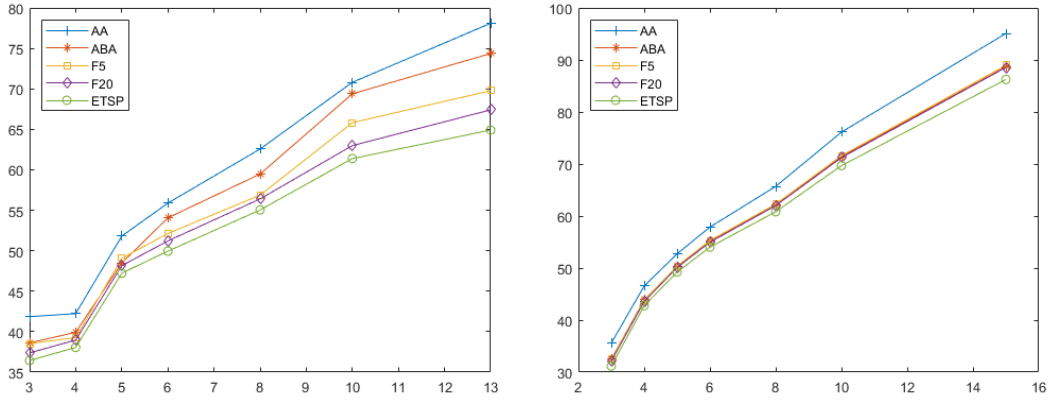


Figura 3.3: A sinistra i risultati ottenuti senza imporre nessun vincolo sulla distanza tra i punti mentre a destra i risultati ottenuti considerando istanze in cui tutti i punti distano almeno $4r$ fra loro. Sull'asse x la grandezza N dei problemi mentre sull'asse delle y la relativa lunghezza dei cammini.

Chiarifichiamo prima di tutto la notazione dei grafici: con F5 e F20 si fa riferimento all'algoritmo di Frego et al. utilizzato sempre con due raffinamenti consecutivi dove i valori 5 e 20 indicano il numero di discretizzazioni ad ogni passaggio. Valori alla mano saltano subito all'occhio alcune evidenze. Come abbastanza prevedibile AA è l'algoritmo meno performante e fornisce sempre le prestazioni peggiori rimanendo dominato da tutte le altre tecniche. Ora vale però la pena fare qualche considerazione in merito a F5 e ABA; come si evince dall'osservazione dei grafici i risultati delle due tecniche sono molto simili, anzi curiosamente nonostante F5 domini ABA nel primo scenario succede il contrario nel secondo. La spiegazione è presto data infatti ABA sfrutta la proprietà osservata nella Proposizione 2.1.2 che quindi si rivela più efficace quando i punti sono distanti tra loro. Quando invece le distanze si accorciano è l'algoritmo di Frego et al. ad avere la meglio poiché discretizzando riesce a gestire in maniera migliore le situazioni in cui sono presenti tratti del tipo RLR e LRL. I risultati ottenuti da F20 dominano ovviamente quelli di F5 ma anche quelli ottenuti da tutti gli altri algoritmi. Sottolineiamo il fatto che il valore 20 è stato scelto in maniera ragionevole confrontando risultati e tem-

pi di calcolo con valori più grandi come 30, 40 e 50. Al crescere del numero di discretizzazioni segue un enorme incremento del tempo di calcolo dal quale però si ottengono risultati quasi identici pertanto la scelta del valore 20 si è rivelata ottimale nell'ottica risultati/tempi.

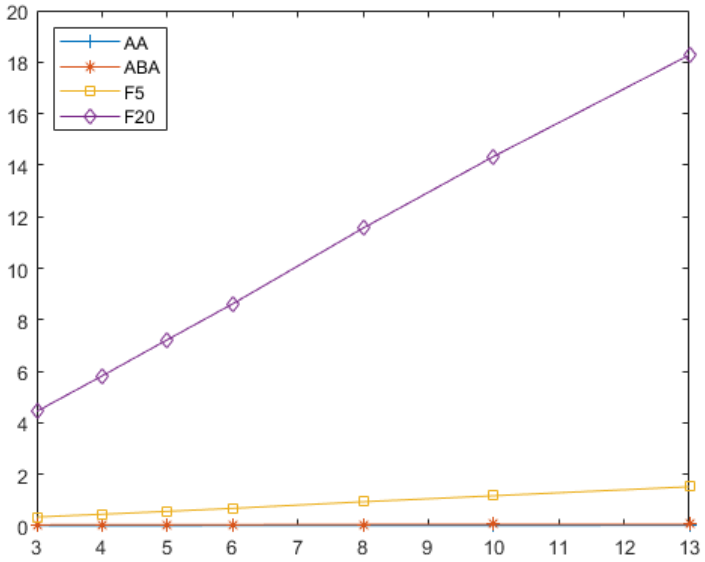
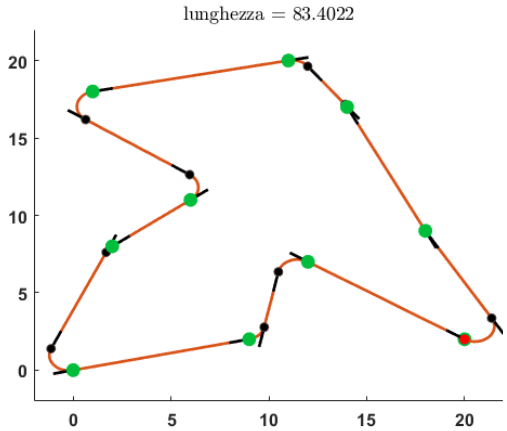
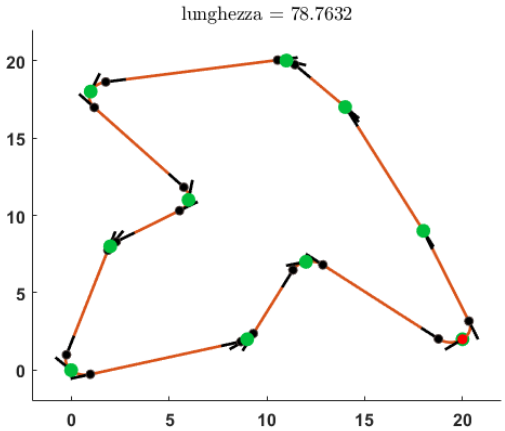


Figura 3.4: Tempi di calcolo espressi in secondi dei vari algoritmi eseguiti in Matlab. Sull'asse delle x la grandezza N del problema.

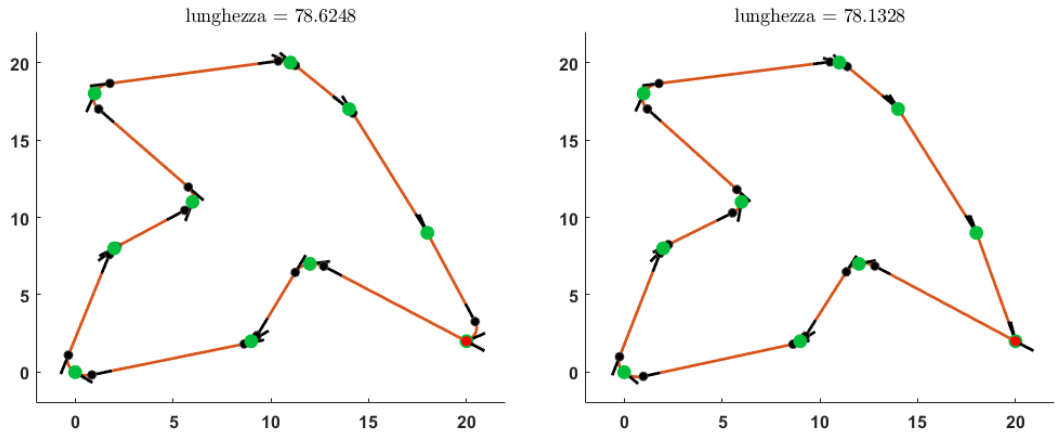
È evidente come gli algoritmi di Frego et al. siano i maggiormente influenzati dalla dimensione dell'istanza. Come si può facilmente dedurre dal grafico i tempi di calcolo crescono significativamente al crescere di N raggiungendo presto valori proibitivi. I tempi di calcolo di ABA e AA oltre a rimanere grossomodo costanti sono decisamente inferiori e trascurabili rispetto a quelli degli altri algoritmi.



Cammino ottenuto con AA



Cammino ottenuto con ABA



Cammino ottenuto con F5 Cammino ottenuto con F20

Figura 3.5: Esempi di cammini ottenuti su un'istanza casuale con le varie tecniche mostrate

In generale possiamo considerare i risultati soddisfacenti nel senso che il gap tra le soluzioni ottenute e i lower bound (che sappiamo essere strettamente minori della soluzioni ottime) è contenuto. Ciò significa che le soluzioni ottenute, per quanto possano non essere ottime, sono sicuramente più che accettabili e ci aspettiamo che almeno in una parte dei casi approssimino fedelmente l'ottimo con errori del tutto trascurabili.

Prima di passare all'analisi di nuove tecniche vale la pena soffermarsi ulteriormente sulle differenze tra cammini di Dubins e cammini euclidei. Per fare ciò proviamo ad analizzare una situazione del tutto simile a quella precedente. Per ogni istanza, scelta ancora una volta in maniera casuale all'interno del quadrato $[0,20]\times[0,20]$, studiamo il comportamento delle soluzioni ottenute al variare del raggio di curvatura minimo. Questo non sarà più fissato al valore $r = 1$ ma verranno studiati e messi a confronto anche i casi in cui $r = 0.1$, $r = 0.5$ e $r = 2$. Nella seguente tabella sono riportati i rapporti tra l'ottimo individuato mediante le varie euristiche e il lower bound fornito dall'ETSP. Come abbiamo già osservato, dal fatto che ogni problema può essere ricondotto ad avere raggio di curvatura unitario, modificare il raggio di curvatura può essere considerato equivalente all'allontanare o avvicinare i punti i tra loro. Da ciò la prima banale osservazione è il fatto che al decrescere del raggio di curvatura ETSP e DTSP tendono ad equivalersi così come evidenziato nella colonna della tabella relativa a $r = 0.1$. In questi casi, e quindi in generale quando i punti sono molto distanti tra loro tutti i metodi forniscono dei buoni risultati. Più interessante è però osservare quello che succede alle soluzioni quando ad una stessa istanza viene modificato il raggio di curvatura. Ovviamente il valore del rapporto aumenta, inoltre si osservano, similmente ai casi precedenti, dei differenti comportamenti dai vari algoritmi. Risulta ancor più evidente il fatto che l'algoritmo di Frego et al. con un'adeguata finezza nella discretizzazione sia l'euristica in grado di ottenere i risultati migliori. Ancora una volta si può notare come ABA risulti un ottimo compromesso tra semplicità e risultati ottenuti ma solamente quando il raggio di curvatura è abbastanza piccolo rispetto alla distanza tra i punti, poiché

in caso contrario è numericamente evidente il crollo delle prestazioni.

N	ALGORITMO	$r = 0.1$	$r = 0.5$	$r = 1$	$r = 2$
4	AA	1.0083	1.0425	1.0881	1.2172
	ABA	1.0025	1.0129	1.0266	1.1564
	FREGO5	1.0022	1.0111	1.0233	1.0543
	FREGO20	1.0017	1.0087	1.0183	1.0410
6	AA	1.0089	1.0476	1.1401	1.3581
	ABA	1.0027	1.0144	1.0571	1.3201
	FREGO5	1.0028	1.0158	1.0346	1.1422
	FREGO20	1.0022	1.0116	1.0248	1.0589
8	AA	1.0079	1.0766	1.1766	1.4625
	ABA	1.0023	1.0126	1.1724	1.4262
	FREGO5	1.0027	1.0147	1.1032	1.3329
	FREGO20	1.0022	1.0118	1.0268	1.0746
10	AA	1.0043	1.0229	1.0711	1.4358
	ABA	1.0012	1.0062	1.0465	1.4290
	FREGO5	1.0014	1.0074	1.0163	1.0703
	FREGO20	1.0010	1.0053	1.0112	1.0486
12	AA	1.0095	1.0512	1.1637	1.5248
	ABA	1.0029	1.0155	1.0953	1.7157
	FREGO5	1.0026	1.0139	1.0324	1.2655
	FREGO20	1.0022	1.0118	1.0260	1.1261

Figura 3.6: Confronto dei risultati ottenuti al variare della dimensione delle istanze e del raggio di curvatura. I valori in tabella sono il rapporto fra il risultato dell'euristica con l'ottimo dell'ETSP associato.

Varianti e migliorie

Fino ad ora il lower bound utilizzato è stato il valore ottimo dell'ETSP, che è facilmente ottenibile e in generale più che accettabile. Vale la pena osservare che tale bound è però migliorabile e una tecnica adeguata per farlo consiste nel rilassare il problema di Dubins al 'Dubins Interval Problem' [21]. I risultati di questa tecnica sono migliori rispetto a quelli ottenuti dall'ETSP ma di certo meno immediati da ottenere.
Richiamando invece la possibilità di fare uso dell'Angular Travelling Salesman Problem per individuare l'ordine dei punti da visitare, riportiamo in breve i risultati ottenuti in [22]. Le varie istanze sono formate da 20 e 21 punti scelti in maniera casuale nel quadrato $[0,500]\times[0,500]$ con raggio di curvatura $r = 50$. Con EAA e AngAA si fa riferimento all'esecuzione dell'Alternating Algorithm eseguito sull'ordinamento scelto rispettivamente con ETSP e AngTSP. ED e AngD fanno invece riferimento ad un algoritmo di discretizzazione risolto all'ottimo tramite l'algoritmo di Dijkstra eseguito ancora una volta sugli ordinamenti ottenuti rispettivamente con ETSP e AngTSP. Nelle tabelle si mostrano i risultati ottenuti con 40 discretizzazioni dell'angolo giro, mentre negli utilizzi dell'Angular TSP i pesi w_1 e w_2 assumono rispettivamente i valori 1 e r.

	Dubins' trajectory length				Ratio with lower bound			
Instance	**EAA**	**ED**	**AngAA**	**AngD**	**EAA**	**ED**	**AngAA**	**AngD**
1	4305,85	2688,15	4357,87	**2462,35**	2,17	1,36	2,20	1,24
2	4231,56	2698,42	3682,61	**2628,88**	2,15	1,37	1,87	1,34
3	3949,40	2628,16	3949,82	**2624,98**	2,54	1,69	2,54	1,69
4	4469,56	**2647,04**	3657,88	2770,28	2,24	1,33	1,84	1,39
5	4135,17	3173,63	3571,63	**2876,89**	1,99	1,52	1,72	1,38
6	4027,10	**2570,08**	3712,75	2759,25	2,27	1,45	2,09	1,56
7	4413,45	**3195,50**	4118,16	3197,21	2,23	1,62	2,09	1,62
8	4047,18	**2805,82**	4000,08	**2805,82**	1,88	1,30	1,86	1,30
9	3862,22	2902,28	4252,23	**2701,50**	2,39	1,80	2,63	1,67
10	4033,08	3030,89	3956,10	**2965,58**	2,12	1,59	2,08	1,56
11	4553,01	2256,05	4494,25	**2180,05**	2,31	1,15	2,28	1,11
12	4325,40	3117,82	3821,44	**3117,75**	2,18	1,57	1,92	1,57
13	4547,60	3240,19	3801,51	**3143,34**	2,16	1,54	1,80	1,49
14	4534,95	3064,07	4041,18	**2856,48**	2,12	1,44	1,89	1,34
15	5004,55	2899,85	3825,48	**2510,49**	2,34	1,36	1,79	1,17
16	3565,93	2594,33	3230,83	**2395,70**	2,05	1,49	1,85	1,37
17	4394,56	3007,35	3774,31	**2681,42**	2,43	1,66	2,08	1,48
18	4503,15	3164,34	4383,80	**3034,21**	2,31	1,63	2,25	1,56
19	4563,84	**2773,08**	4563,84	**2773,08**	2,50	1,52	2,50	1,52
20	4460,31	3307,51	3718,95	**2901,43**	2,10	1,56	1,75	1,37
				Average	2,22	1,50	2,05	1,44

Figura 3.7: In grassetto i risultati migliori ottenuti su ogni istanza. Nella parte di destra è mostrato il rapporto tra il valore ottimo trovato e il lower bound.

Dai risultati numerici si evince ancora una volta come gli algoritmi di discretizzazione si rivelino particolarmente efficienti; è inoltre possibile osservare che, soprattutto nel caso in cui non ci siano vincoli sulla distanza tra punti, può essere determinante scegliere un ordinamento differente da quello dell'ETSP. Di seguito la rappresentazione grafica dell'istanza 11 in cui l'ordinamento fornito dall'AngTSP è più efficace rispetto a quello ottenuto dall'ETSP.

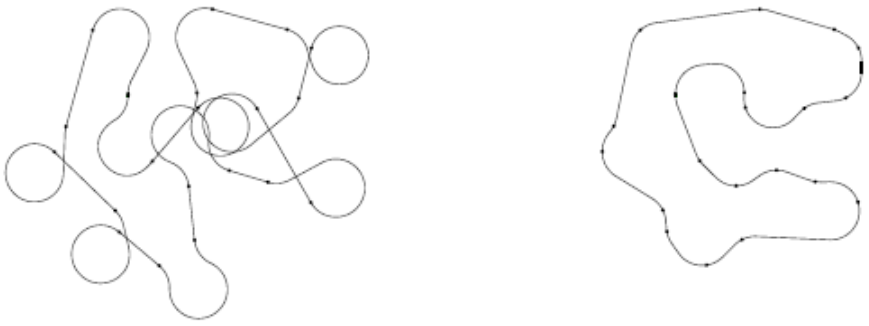
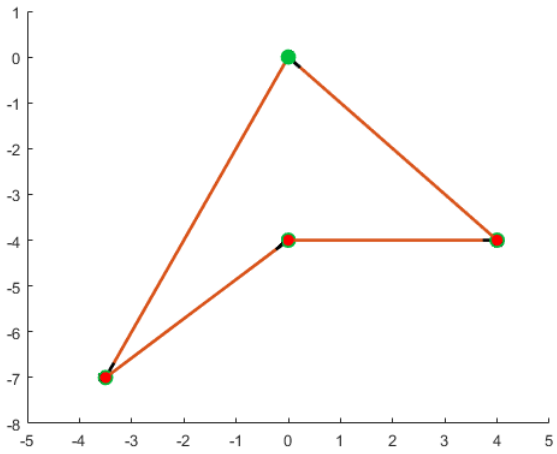


Figura 3.8: A sinistra il risultato ottenuto con EAA mentre a destra quello ottenuto con AngD.

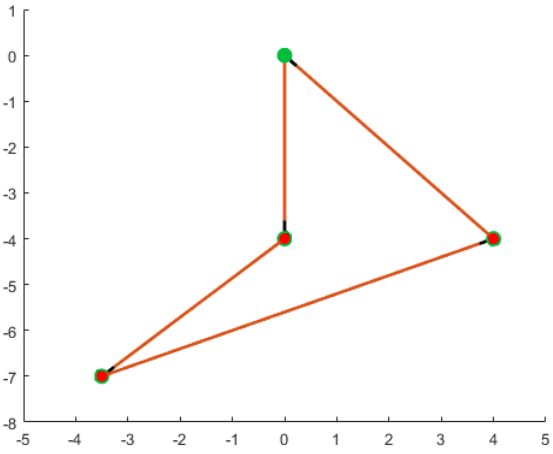
3.3 Algoritmi senza utilizzo dell'ETSP

Per quanto le tecniche euristiche mostrate siano in grado di ottenere anche degli ottimi risultati, è evidente che tali approcci non siano sufficienti per garantire l'ottimalità delle soluzioni. Come si può aver intuito dai risultati precedenti l'ordinamento di ETSP e DTSP potrebbero non coincidere. Questo accade con maggiore frequenza nelle istanze i cui punti sono in posizioni ravvicinate, ma ciò può accadere anche quando la distanza tra punti è superiore a $4r$ come testimonia l'esempio seguente. Di conseguenza, anche se spesso e volentieri le sequenze coincidono, in generale non può essere considerata una regola.

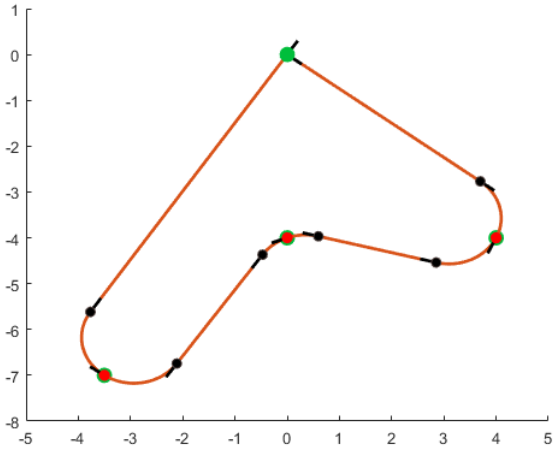
Esempio: consideriamo il cammino che inizia e termina in $u_0 = (0, 0)$ e che deve visitare l'insieme di punti $u_1 = (4, -4), u_2 = (0, -4)$ e $u_3 = (-\frac{7}{2}, -7)$ e mettiamo a confronto due differenti sequenze. Siano le due sequenze $u_0u_1u_2u_3u_0$ e $u_0u_1u_3u_2u_0$, possiamo osservare che la prima è quella con ETSP ottimo mentre la seconda, per quanto sia più lunga per la distanza euclidea possiede il DTSP ottimo.



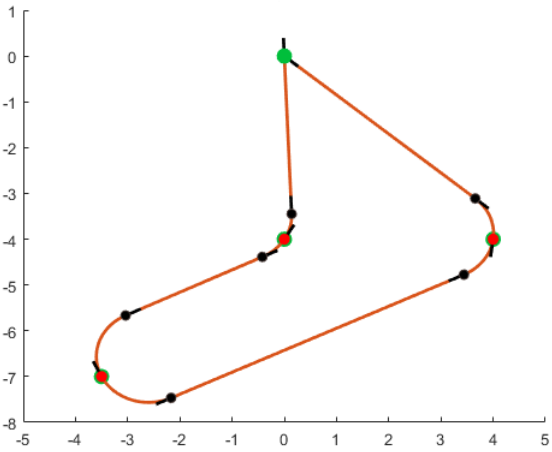
Lunghezza = 22.0929



Lunghezza = 22.3444



Lunghezza = 23.6845



Lunghezza = 23.6668

È evidente dall'esempio mostrato che la differente disposizione dei tratti curvi influenza in maniera diversa la lunghezza del cammino di Dubins totale. Non è

particolarmente raro trovarsi di fronte a casi del genere e di conseguenza adottare la sequenza ottima dell'ETSP non è per nulla garanzia di successo pertanto è necessario adottare delle nuove strategie.

3.3.1 Algoritmo di Dhulipala et al.

La prima tecnica che non si basa sulla sequenza dell'ETSP che presentiamo è in realtà un'euristica che prende spunto dalla classica tecnica di programmazione dinamica per la risoluzione del TSP su un grafo e viene adattata ai cammini di Dubins. L'algoritmo sviluppato da Dhulipala et al. [8] sfrutta la ricorsività del problema facendo uso della distanza di Dubins rilassata.
Sia $\{u_0, u_1, ..., u_N\}$ un insieme di punti con u_0 il punto di partenza e c_{ij} la lunghezza del tratto che va dal punto u_i al punto u_j. Dato un insieme $S \subset \{u_1, ..., u_N\}$ e un punto $u_l \in S$ definiamo $C(S, l)$ la lunghezza del cammino più breve che comincia in u_0, visita tutti i punti dell'insieme S terminando in u_l. In generale valgono le seguenti regole ricorsive:

- $C(\{u_l\}, l) = c_{0l}, l \in \{u_1, ..., u_N\}$ per $|S| = 1$

- $C(S, l) = \min_{m \in S - \{u_l\}} [C(S - \{u_l\}, m) + c_{ml}]$ per $|S| > 1$

A questo punto il cammino più breve è $\mathcal{C} = \min_{l \in \{u_1, ..., u_N\}} [C(\{u_1, ..., u_N\}, m) + c_{l0}]$, mentre la sequenza può essere ricavata a ritroso. L'algoritmo ha complessità esponenziale e assume che i valori c_{ij} per $i \neq j$ siano noti a priori.
Cerchiamo adesso di capire come modificare l'algoritmo in modo che risolva le istanze del DTSP. È doveroso evidenziare che anche in questo caso, nonostante gli ottimi risultati che si possono ottenere, il metodo è euristico e non dà in alcun modo garanzie di ottimalità. È inoltre necessario essere a conoscenza della direzione di partenza nel punto u_0.
Come prima cosa si calcolano il costi c_{0j} della distanza di Dubins rilassata tra il punto u_0 e il punto u_j con $j = 1, ..., N$ (per cammino di Dubins rilassato si intende quel cammino in cui è libera la direzione di arrivo). Successivamente si calcolano, quando possibile, ovvero quando si è a conoscenza della direzione in u_i, i valori c_{ij}. Denotiamo inoltre con $\alpha(S, l)$ l'angolo in $u_l \in S$ ottenuto dopo aver visitato tutti i punti di S ed aver terminato il cammino in u_l. Per ogni $C(S, l)$ teniamo in memoria u_l e $\alpha(S, l)$. Il cammino risultante sarà una concatenazione di cammini di Dubins rilassati. Si veda l'Appendice B per lo pseudocodice dell'algoritmo.

Osservazione: Come si può aver facilmente notato, le tecniche al momento prese in analisi sono solamente delle euristiche. Vista la complessità del DTSP, il problema è stato affrontato in letteratura solamente con delle tecniche di approssimazione. Per chi è invece alla ricerca del valore ottimo esatto, proponiamo una tecnica in grado di risolvere il problema nel caso in cui la distanza tra i punti è superiore a $4r$.

3.3.2 Algoritmo di programmazione dinamica

Per raggiungere l'obiettivo dell'ottimalità facciamo uso della programmazione dinamica; è possibile infatti estendere i risultati ottenuti nel paragrafo 2.5 facendo uso del quinto modello PNL e adattando adeguatamente le procedure dell'Algoritmo 1.
In questa estensione l'etichetta di ogni stato è definita da una terna $\{i, S, \sigma_i\}$ dove:

- i è l'indice dell'ultimo punto sorvolato u_i;
- S è l'insieme ordinato degli indici dei punti sorvolati, i incluso;
- $\sigma_i \in \{-1, 1\}$ direzione di sorvolo del punto u_i.

Ad ogni stato è associato il vettore σ che contiene le direzioni $\{-1, 1\}$ di sorvolo fino al punto i. Dal generico stato $\mathcal{L} = \{i, S, \sigma_i\}$ si genera lo stato $\mathcal{L}' = \{j, S \cup \{j\}, \sigma_j\}$ tale per cui $j \notin S$.
L'idea è quella di seguire una procedura che combina la classica programmazione dinamica per il TSP con l'approccio presentato nel capitolo precedente. Risulta evidente che il carico computazionale che ne deriva è molto elevato, la diretta conseguenza è che è possibile approcciare solamente problemi di piccole dimensioni. È importante evidenziare un'ulteriore difficoltà che si presenta nella risoluzione del DTSP: se nel classico algoritmo di Held e Karp per la risoluzione del TSP è possibile identificare la lunghezza di un cammino che passa per i punti a partire dalla lunghezza del cammino formato da $i-1$ punti, questo non è più possibile nel caso dei cammini di Dubins. I vincoli cinematici anolonomi sono tali per cui natura la ricorsiva del problema viene meno (salvo discretizzazioni). Di conseguenza ogni volta che si aggiunge un nuovo punto ad una sequenza è necessario ricalcolare la lunghezza totale del cammino.
Prima di presentare lo psudocodice dell'Algoritmo 2 facciamo qualche considerazione circa la notazione adottata. Le funzioni *Modello* e *Ammissibilità* sono analoghe a quelle dell'Algoritmo 1; variano invece i test di verifica della possibile ottimalità. Se nel caso della sequenza prefissata è sufficiente confrontare la lunghezza dei cammini che arrivano ad un determinato punto u_i ora è necessario verificare anche quali punti sono stati visitati prima di giungere proprio in u_i. Di conseguenza il test di dominanza presentato nel capitolo precedente va sistemato tenendo in considerazione l'insieme S di punti visitati durante il cammino in questione. Dati due differenti stati $\mathcal{L}'$ e $\mathcal{L}''$ con uguale indice i, possiamo eliminare (dominandolo) quello di lunghezza maggiore (in questo caso $\mathcal{L}''$) se valgono le due condizioni $c(\mathcal{L}'') - c(\mathcal{L}') \geq \frac{7}{3}\pi - 2(1+\sqrt{3})$ e $S'' \subseteq S'$. Vale anche in questo caso il rafforzamento del bound della prima condizione.

Algoritmo 2 PROGRAMMAZIONE DINAMICA PER IL DTSP

DATI: $\overline{S} = \{u_0, ..., u_N\}$

for $i \in 1, .., N$ **do**
S = $\{u_i\}$;
Risolvi(Modello(i,S));
Modello(i,S,R) = Modello(i,S,L) = Modello(i,S);
end for
while esiste un cammino aperto **c** con insieme S $\neq \overline{S}$ **do**
Scegli $u_j \in \overline{S} \backslash$S
Espandi il cammino aperto **c**
S = S $\cup\{u_j\}$;
r_1 = Risolvi(Ammissibilità(j,S,R));
r_2 = Risolvi(Ammissibilità(j,S,L));
if $r_1 = r_2 = 1$ **then**
Risolvi(Modello(j,S));
Modello(j,S,R) = Modello(j,S,L) = Modello(j,S);
Testa le possibili ottimalità ed eventualmente **scarta** i cammini
else if $r_1 = 1$ e $r_2 > 1$ **then**
Scarta il cammino con la sequenza attuale + (j, L)
Risolvi(Modello(j,S,R));
Testa la possibile ottimalità ed eventualmente **scarta** il cammino
else if $r_1 > 1$ e $r_2 = 1$ **then**
Scarta il cammino con la sequenza attuale + (j, R)
Risolvi(Modello(j,S,L));
Testa la possibile ottimalità ed eventualmente **scarta** il cammino
else if $r_1 > 1$ e $r_2 > 1$ **then**
Scarta il cammino con la sequenza attuale
end if
end while
while esiste un cammino aperto **c** **do**
Scegli u_j t.c. $\overline{S}$=S
Espandi il cammino aperto **c**
r_1 = Risolvi(Ammissibilità(j,S,R));
r_2 = Risolvi(Ammissibilità(j,S,L));
if $r_1 = r_2 = 1$ **then**
Risolvi(Modello(N));
Se è il risultato migliore ottenuto fino ad ora **MIN** = Risolvi(Modello(j,S))
end if
end while
return MIN

Possibili migliorie

Come già più volte fatto notare, la complessità dell'algoritmo appena proposto è esponenziale, il che lo rende utilizzabile solamente su istanze piccole (non oltre i 9-10 punti). Per istanze più grosse vale la pena adottare altre tecniche più performanti. Ultimo, ma non per importanza, vale la pena mettere in luce qualche possibile miglioria in grado di velocizzare l'algoritmo. Innanzitutto può fare comodo avere a disposizione una soluzione ammissibile per cercare di capire se un determinato stato ancora aperto può portarci ad una soluzione migliore. Per fare ciò è sufficiente utilizzare un'euristica di quelle presentate nel paragrafo precedente in maniera tale da avere a disposizione una soluzione ammissibile accettabile con la quale confrontare i possibili risultati. Dato un cammino parziale è necessario completarlo con la linea spezzata ottima e successivamente fare il confronto con la migliore soluzione ammissibile che abbiamo a disposizione. Ovviamente se il cammino parziale completato dalla spezzata supera in lunghezza la soluzione ammissibile possiamo scartare il relativo stato.
L'altra possibilità per incrementare la velocità di esecuzione dell'algoritmo risiede nel possibile approccio bidirezionale. Sia nel caso in cui il punto di partenza che quello di arrivo coincidano che nel caso in cui i due punti siano distinti, un approccio di questo tipo può rivelarsi efficace. Come già noto in letteratura [3] [30] [28] questa tecnica permette generalmente di ottenere risultati migliori della semplice ricerca monodirezionale. L'idea di fondo risiede nel fatto di esplorare i cammini a partire sia dal punto di partenza che da quello di arrivo cercando poi di collegarli, tendenzialmente intorno a metà strada. Nel nostro caso, poiché partenza ed arrivo coincidono, vogliamo usare questa strategia per collegare due cammini parziali disgiunti e formare il cammino ottimo intero. Ancora una volta però è necessario constatare una problematica che si è già presentata più volte nel corso del lavoro, come infatti abbiamo più volte potuto osservare in precedenza non è possibile collegare due istanze per via della non ricorsività del problema continuo e quindi l'approccio bidirezionale può fungere solamente da bound per individuare i potenziali candidati ottimi. L'idea è pertanto quella di espandere la ricerca dell'ottimo in ampiezza procedendo come illustrato precedentemente e successivamente verificare se l'unione di due sottopolitiche può generare un possibile ottimo. Per fare ciò è necessario congiungere due sottocammini disgiunti e complementari che giungono ad uno stesso punto. Se la somma delle lunghezze dei sottocammini supera la lunghezza del miglior cammino ammissibile a disposizione, le sottopolitiche possono essere eliminate.

Risultati e tempi di calcolo

Come più volte segnalato la complessità del problema è elevata. Il solo algoritmo di Held e Karp ha complessità temporale $\Theta(2^N N^2)$ a cui va aggiunto che nel nostro caso ogni nodo aggiunto va visitato con entrambe le direzioni di passaggio R e L. In aggiunta a ciò ci sarà bisogno di conservare i dati necessari all'inizializzazione oppure rigenerarli risolvendo in sequenza i modelli PNL. Ricordiamo inoltre che all'aggiunta di un nuovo punto alla sequenza è necessario testarne, a destra e a sinistra, l'ammissibilità. Per dare un'idea dei tempi di calcolo, fissando gli errori di ammissibilità a 10^{-8} i solutori impiegano tempi tra gli 0.01 e gli 0.5 secondi per individuare la soluzione. Queste abissali differenze dipendono dalla bontà dell'ini-

zializzazione; come abbiamo potuto osservare nei test effettuati, ottime inizializzazioni permettono una convergenza quasi immediata all'ottimo. Quando invece si adotta un approccio multistart favorendo l'inizializzazione casuale i tempi di calcolo lievitano considerevolmente. È chiaramente presente una dipendenza dei tempi di calcolo rispetto alle dimensioni delle istanze ma questo divario non risulta così evidente vista la limitatezza delle dimensioni nei test unita alla variabilità dei tempi dovuta alle inizializzazioni.

Capitolo 4

Conclusioni

4.1 Risultati ottenuti

In questo lavoro sono stati trattati alcuni algoritmi di routing per veicoli anolonomi. Fin da subito abbiamo potuto cogliere come tali vincoli cinematici modifichino notevolmente le traiettorie ottime dei veicoli che ne sono sottoposti. Inizialmente abbiamo richiamato gli ormai più che noti risultati ottenuti da L. Dubins [9] per la ricerca del cammino minimo tra due punti con le direzioni fissate. Il primo risultato importante in cui ci si imbatte afferma che quando un veicolo ha la possibilità di muoversi solamente in una direzione con curvatura limitata, il tragitto più breve per connettere due configurazioni nello spazio è composto solamente da tratti rettilinei e archi di circonferenza con raggio di curvatura minimo. Abbiamo così potuto discutere le caratteristiche della distanza di Dubins che sta alla base di tutte le geometrie dei cammini in cui ci si imbatte nel corso del lavoro. A partire dai risultati mostrati in questa prima parte abbiamo poi ampliato il problema imponendo il passaggio per un numero di punti superiore a due cercando di affrontare le naturali difficoltà in cui ci si imbatte. Le prime difficoltà che sono spontaneamente sorte derivano dalla non ricorsività del problema continuo. Se ricavare il cammino ottimo tra due punti consecutivi non crea intoppi, non si può dire lo stesso quando il numero di punti da raggiungere aumenta. Contrariamente a quanto si possa pensare di primo acchito, non è sufficiente concatenare una serie di soluzioni ottime tra due punti ma bisogna valutare il problema nella sua interezza. Ogni punto da visitare che si aggiunge ad una sequenza già nota può modificare, anche significativamente, l'intero cammino. Questo come abbiamo visto crea non pochi grattacapi, infatti se quando i punti sono solamente due è sufficiente confrontare le possibili configurazioni ottime (si veda il Teorema 1.2.1), è impensabile fare lo stesso quando il problema diventa grande. Le possibili strategie per individuare il cammino minimo in questi casi sono molteplici. La prima idea che sorge in maniera abbastanza naturale è quella di costruire un modello di programmazione matematica (non lineare) del problema basandosi sulle caratteristiche geometriche del cammino. La soluzione al problema in questi casi deriva dalla risoluzione, tramite adeguati solutori, dei modelli costruiti. Come abbiamo però potuto osservare questa tecnica non si rivela sempre efficace per via dei frequenti ottimi locali che vengono individuati dai solutori numerici. La prima proposta che abbiamo avanzato per ovviare il problema è valida nel caso in cui i punti successivi distino tra loro più di $4r$, in questi casi la soluzione ottima ha delle caratteristiche ben precise che ci permettono di riformulare

i modelli matematici. In questi modelli è necessario conoscere l'orientamento con cui vengono percorsi i tratti curvi; se non è noto a priori va individuato grazie ad un algoritmo di programmazione dinamica che confronta le differenti scelte discrete scegliendo le migliori.
L'altra tecnica presentata nella tesi, efficace anche quando i punti da raggiungere sono vicini tra loro, è la discretizzazione. L'idea che sta alla base della discretizzazione è quella di rendere in numero finito gli angoli di passaggio. Ciò semplifica notevolmente il problema e come osservato nell'algoritmo di Frego et al. [10] ci permette di adottare un approccio ricorsivo. In questo modo il cammino minimo può essere identificato senza l'ausilio di solutori numerici. Il prezzo da pagare è che la soluzione che ne deriva risulta approssimata, anche se come abbiamo già potuto notare questo non è un grosso ostacolo poiché i valori approssimati osservati nei nostri test si sono rivelati molto accurati e di poco scostati dall'ottimo. È bene tuttavia evidenziare il fatto che l'algoritmo rimane un'euristica che non dà nessuna garanzia sul fatto di trovare le corrette configurazioni dell'ottimo. Una volta descritte e analizzate queste tecniche il lavoro si è focalizzato sull'ultima fase, ovvero l'individuazione del cammino ottimo per un insieme di punti. Qui non è più sufficiente trovare i corretti orientamenti di passaggio poiché è necessario identificare anche la sequenza di passaggio corretta. Come abbiamo potuto osservare nel corso del terzo capitolo della tesi i metodi per fronteggiare quest'ultimo problema sono principalmente due. Una prima strategia è quella di scegliere la sequenza di punti da raggiungere individuando il cammino ottimo ottenuto con la distanza euclidea e solo successivamente scegliere gli orientamenti di passaggio. Abbiamo così potuto presentare alcune tecniche euristiche adatte ad una rapida risoluzione del DTSP (Dubins TSP) senza che venga in nessun modo garantita l'ottimalità del risultato. A questo proposito è necessario richiamare un'importante osservazione già fatta nel corso del lavoro, ovvero che la sequenza ottima ottenuta con la distanza euclidea non sempre coincide con quella ottima ottenuta con la distanza di Dubins e quindi se si vuole ottenere l'ottimo bisogna fare affidamento ad altre tecniche. Quello che infatti prevede la seconda delle strategie adottate è che non venga scelta a priori la sequenza da percorrere ma che questa venga scoperta in itinere. Oltre all'algoritmo euristico di Dhulipala et al. [8] abbiamo proposto, nel caso in cui la distanza tra i punti è superiore a $4r$, un algoritmo di programmazione dinamica estensione di quello proposto per le sequenze di punti in grado di individuare l'ottimo.
È opportuno fare qualche considerazione conclusiva circa il lavoro svolto. Come già abbiamo fatto notare nel preambolo, l'argomento affrontato sta raccogliendo negli anni un crescente interesse per via dei suoi risvolti pratici. Nonostante ciò, allo stato dell'arte sono le euristiche a farla da padrone poiché ottenere le soluzioni ottime è alquanto complicato e costoso (computazionalmente parlando). Abbiamo cercato di presentare gli algoritmi più significativi presenti in letteratura e a partire da essi abbiamo sviluppato dei nuovi modelli adatti a risolvere alcune istanze. Quello che si evince dal lavoro svolto è il fatto che non esiste una tecnica risolutiva del problema migliore a priori. Quanto appena detto va inteso nel senso che ogni particolare applicazione dei cammini di Markov-Dubins o del DTSP deve essere approcciata in funzione dei risultati che si necessita di ottenere. Se l'obiettivo finale è quello di calcolare un percorso accettabile che si avvicini abbastanza alla soluzione ottima può essere sufficiente fare affidamento agli algoritmi euristici che come abbiamo visto si rivelano spesso molto efficaci e rapidi. Merita ancora una volta una menzione speciale l'algoritmo di Frego et al. [10] presentato nel Paragrafo 2.6.1 che come

rapporto complessità risultati ha fornito le migliori performance ottenute tramite euristiche. Se invece siamo alla ricerca dell'ottimo numerico esatto, ad oggi le uniche possibilità risiedono nell'utilizzo di modelli matematici risolti numericamente all'ottimo. In questi casi bisogna fare i conti con le difficoltà di convergenza che se però vengono affrontate adeguatamente possono essere brillantemente superate. Giunti alle battute conclusive è bene sottolineare un'altra volta che i risultati ottenuti in questo lavoro non sono da intendere come un capitolo a sé stante bensì come fondamenta per futuri approfondimenti ed espansioni. Nel seguito lasciamo alcuni spunti adatti all'ampliamento dei problemi presentati nel corso della tesi.

4.2 Sviluppi ulteriori

Una prima idea per approfondire il lavoro presentato risiede nell'adattare gli algoritmi di programmazione dinamica proposti in maniera tale da prendere in considerazione anche cammini senza vincoli sulla distanza tra punti. Questo potrebbe rivelarsi un problema spinoso perché per farlo è necessario costruire dei nuovi modelli matematici (più complessi) e modificare in maniera sostanziale le tecniche di bounding.
Un'altra importante possibilità risiede nel costruire e risolvere all'ottimo l'Orienteering Problem, le cui istanze fungono da problema di pricing per Vehicle Routing Problem associato. Risolvendo all'ottimo l'OP si è poi in grado tramite generazione di colonne e algoritmi di branch and price di gestire una flotta intera di veicoli anolonomi ottimizzandone i movimenti.

Come ultime proposte lasciamo dei suggerimenti per differenti estensioni dei problemi presentati in cui vengono introdotte delle modifiche per adeguare le istanze ad alcune esigenze reali.

4.2.1 La discontinuità dell'accelerazione

In tutto il lavoro svolto, le geometrie dei cammini ottenuti sono semplici. Abbiamo osservato infatti che tutti i cammini sono composti solamente da tratti rettilinei e archi di circonferenza di raggio r fissato. In questo modo però quello che succede è che l'accelerazione del veicolo, nel cambio tra tratto rettilineo e curvo o tra differenti tratti curvi, è discontinua. L'accelerazione $\vec{a}$ [15] risulta essere scomponibile in due componenti, una tangenziale $\vec{T}(t)$ e una ortogonale $\vec{N}(t)$. Quello che si osserva è che $\vec{a}(t) = a_T\vec{T}(t) + a_N\vec{N}(t)$ con $a_T = \frac{d}{dt}(X'(t)) = X''(t)$ e $a_N = \kappa(X'(t))^2$.
Ricordiamo che la curvatura κ nel caso di un arco di circonferenza di raggio r assume il valore $\kappa = r^{-1}$ mentre nei tratti rettilinei $\kappa = 0$. Nei casi fino ad ora analizzati l'accelerazione tangenziale è nulla in quanto abbiamo supposto che la velocità dei veicoli sia costante. Non possiamo dire la stessa cosa invece per quel che riguarda l'accelerazione ortogonale che, in linea con la curvatura, risulta essere costante a tratti (Figura 4.1). Queste discontinuità generano delle criticità nell'applicazione concreta dei cammini con il rischio che il contraccolpo dovuto all'accelerazione centripeta e alla dispersione dovuta all'impossibilità pratica di applicare istantaneamente una curvatura. Tutto ciò rende complicato seguire in maniera esatta la traiettoria preposta. Per porre rimedio a questo problema è necessario raccordare i tratti dritti con gli archi di circonferenza in modo che la curvatura e di conseguenza

l'accelerazione vari in maniera continua evitando i problemi citati. Per fare ciò si possono raccordare i tratti mediante diversi tipi di curve come le clotoidi (Figura 4.2), le curve di Bézier e le B-splines [42]. In generale si può mostrare che quando due punti consecutivi sono "abbastanza distanti" e collegati da cammini del tipo CSC nel cammino di Dubins ottimo, è facile aggiustare il cammino in modo da renderlo due volte derivabile in ogni punto. La situazione tuttavia è differente quando i punti sono "vicini" tra loro e sarà necessario un tipo di interpolazione diverso.

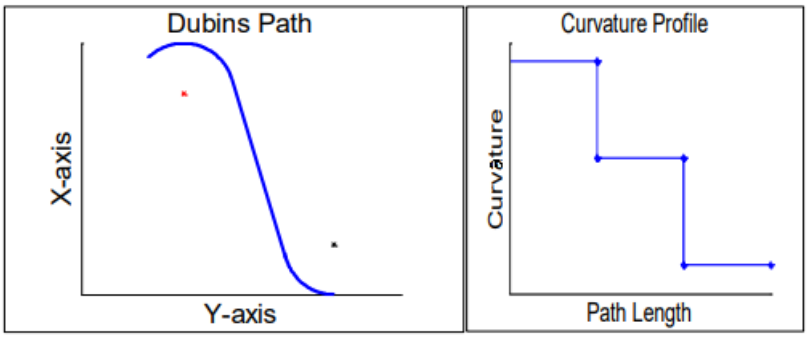


Figura 4.1: Cammino di Dubins e profilo della curvatura [2]

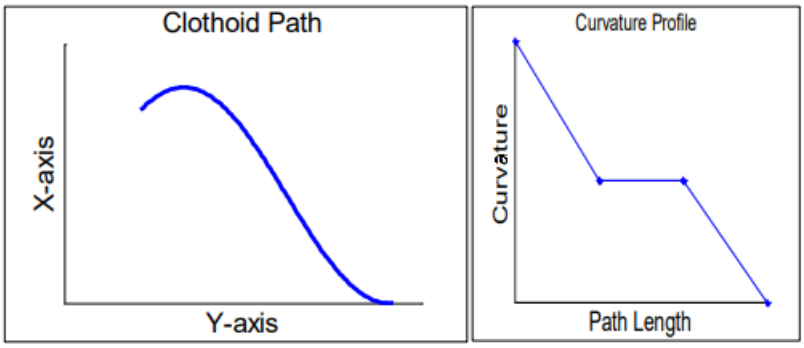


Figura 4.2: Cammino con clotoidi e profilo della curvatura [2]

4.2.2 Il problema tridimensionale, la presenza di ostacoli e l'avvicinamento

Se i veicoli dei quali stiamo approssimando i movimenti sono ad esempio dei droni, viene naturale pensare di estenderci alla terza dimensione spaziale [16]. L'estensione non è per nulla banale e sono possibili differenti approcci per affrontare il problema. Un'altra variante che sorge spontaneamente nella risoluzione di problemi concreti (sia in due che in tre dimensioni), è quella che prevede la necessità di schivare degli ostacoli o delle intere zone presenti nello spazio [20] [43]. Un ultimo spunto che merita di essere nominato è il problema dell'avvicinamento; supponiamo che sia necessario per qualche motivo monitorare una zona e quindi visitare alcuni punti salienti. Potrebbe non essere necessario passare esattamente per i punti assegnati ma solamente in un loro intorno più o meno grande. Nasce così la variante in cui invece di visitare dei punti si visitano degli intorni di essi. Il problema poi può essere affrontato sia con sequenza assegnata, sia con sequenza da stabilire [27] [40].

Appendice A

Esempi di problemi

In questa appendice possiamo trovare alcuni problemi singolo veicolo che sono stati affrontati nella tesi. Alcuni di essi sono stati presi da [18] mentre altri sono stati costruiti ad hoc per verificare particolari caratteristiche sia delle soluzioni che dei solutori numerici utilizzati. Le soluzioni ottime proposte sono rappresentate come per il resto del lavoro con Matlab.

A.1 Problema 1

Dimensione Problema: N = 3
Raggio di curvatura: $r = 1$

$$\begin{cases} (x_0, y_0, \theta_0) & = (0, 0, \frac{\pi}{2}) \\ (x_1, y_1) & = (1, 6) \\ (x_2, y_2) & = (5, 4) \\ (x_3, y_3, \theta_3) & = (3, 0, -\frac{\pi}{2}) \end{cases}$$

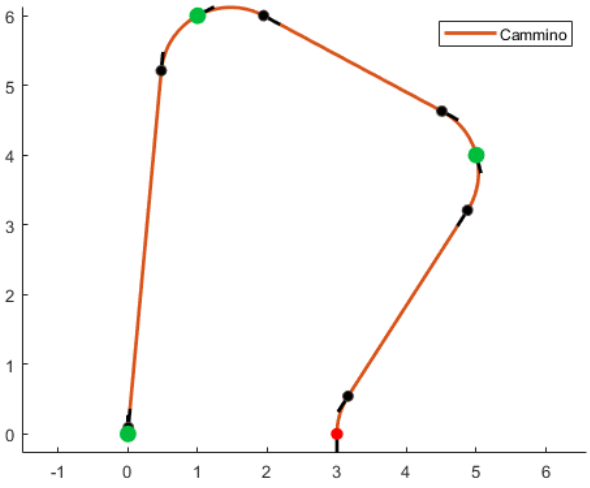


Figura A.1: Cammino ottimo del tipo RSRSRSL con lunghezza 15.4985

A.2 Problema 2

Dimensione Problema: N = 3
Raggio di curvatura: $r = \frac{1}{3}$

$$\begin{cases} (x_0, y_0, \theta_0) & = (0, 0, -\frac{\pi}{3}) \\ (x_1, y_1) & = (-0.1, 0.3) \\ (x_2, y_2) & = (0.2, 0.8) \\ (x_3, y_3, \theta_3) & = (1, 1, -\frac{\pi}{6}) \end{cases}$$

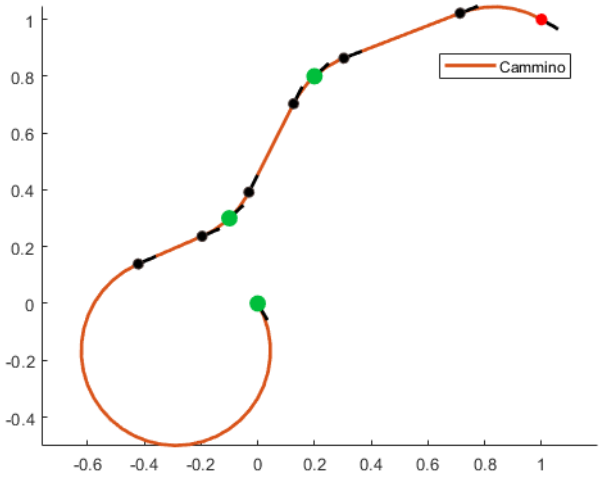


Figura A.2: Cammino ottimo del tipo RSLSRSR con lunghezza 3.4156

A.3 Problema 3

Dimensione Problema: N = 5
Raggio di curvatura: $r = \frac{1}{3}$

$$\begin{cases} (x_0, y_0, \theta_0) & = (0, 0, -\frac{\pi}{3}) \\ (x_1, y_1) & = (-0.1, 0.3) \\ (x_2, y_2) & = (0.2, 0.8) \\ (x_3, y_3) & = (1.0, 1.0) \\ (x_4, y_4) & = (0.5, 0.5) \\ (x_5, y_5, \theta_5) & = (0.5, 0, -\frac{\pi}{6}) \end{cases}$$

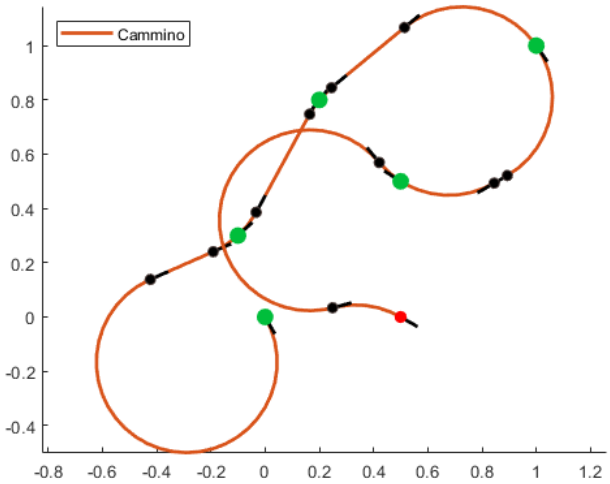


Figura A.3: Cammino ottimo del tipo RSLSRSRSRLR con lunghezza 6.2780

A.4 Problema 4

Dimensione Problema: N = 11
Raggio di curvatura: $r = \frac{1}{3}$

$$\begin{cases} (x_0, y_0, \theta_0) & = (0.5, 1.2, \frac{5}{6}\pi) \\ (x_1, y_1) & = (0.0, 0.5) \\ (x_2, y_2) & = (0.5, 0.5) \\ (x_3, y_3) & = (1.0, 0.5) \\ (x_4, y_4) & = (1.5, 0.5) \\ (x_5, y_5) & = (2.0, 0.5) \end{cases} \quad \begin{cases} (x_6, y_6) & = (2.0, 0.0) \\ (x_7, y_7) & = (1.5, 0.0) \\ (x_8, y_8) & = (1.0, 0.0) \\ (x_9, y_9) & = (0.5, 0.0) \\ (x_{10}, y_{10}) & = (0.0, 0.0) \\ (x_{11}, y_{11}, \theta_{11}) & = (0.0, -0.5, 0) \end{cases}$$

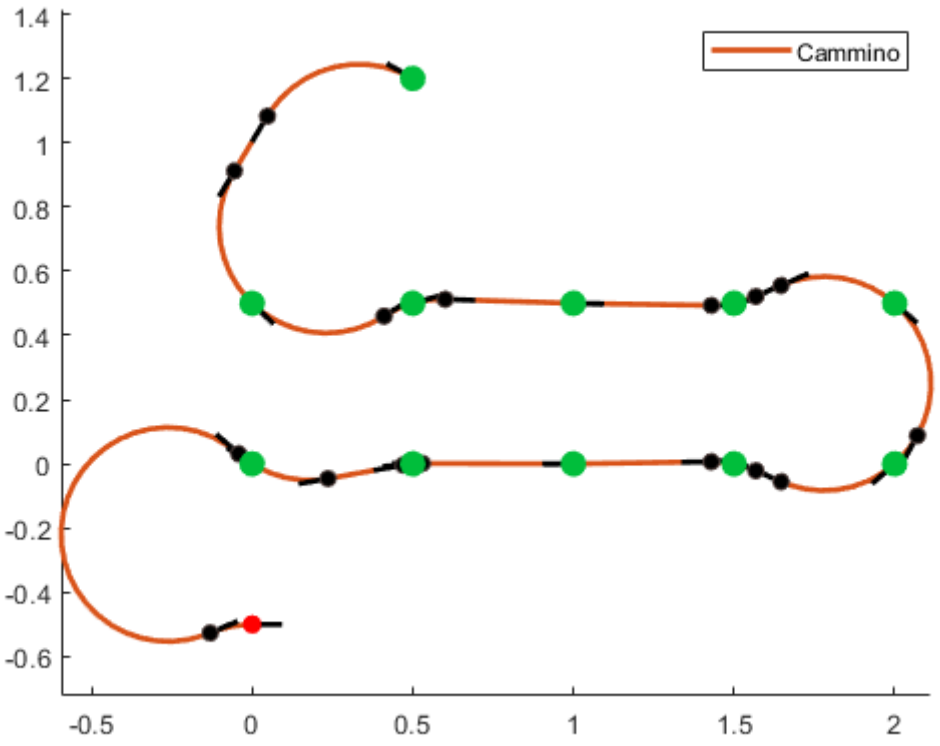


Figura A.4: Cammino ottimo del tipo LSLRSLSLSRSLSRSLSRLR con lunghezza 7.4676

A.5 Problema 5

Dimensione Problema: N = 26
Raggio di curvatura: $r = \frac{1}{3}$

$$\begin{cases} (x_0, y_0, \theta_0) & = (2.9, -1.7, \frac{9}{10}\pi) \\ (x_1, y_1) & = (2.7, -1.7) \\ (x_2, y_2) & = (2.5, -2.4) \\ (x_3, y_3) & = (1.9, -2.6) \\ (x_4, y_4) & = (1.2, -2.6) \\ (x_5, y_5) & = (1.9, -1.2) \\ (x_6, y_6) & = (2.8, 0.92) \\ (x_7, y_7) & = (0.0047, 3.2) \\ (x_8, y_8) & = (-2.8, 0.92) \\ (x_9, y_9) & = (-1.9, -1.2) \\ (x_{10}, y_{10}) & = (-1.2, -2.6) \\ (x_{11}, y_{11}) & = (-1.9, -2.6) \\ (x_{12}, y_{12}) & = (-2.5, -2.4) \\ (x_{13}, y_{13}) & = (-2.7, -1.7) \end{cases}$$

$$\begin{cases} (x_{14}, y_{14}) & = (-2.9, -1.7) \\ (x_{15}, y_{15}) & = (-2.6, -3.2) \\ (x_{16}, y_{16}) & = (-1.1, -3.2) \\ (x_{17}, y_{17}) & = (-0.9, -3) \\ (x_{18}, y_{18}) & = (-1.5, -0.92) \\ (x_{19}, y_{19}) & = (-1.9, 0.93) \\ (x_{20}, y_{20}) & = (-0.0047, 3) \\ (x_{21}, y_{21}) & = (1.9, 0.93) \\ (x_{22}, y_{22}) & = (1.4, -0.92) \\ (x_{23}, y_{23}) & = (0.9, -3) \\ (x_{24}, y_{24}) & = (1.1, -3.2) \\ (x_{25}, y_{25}) & = (2.6, -3.2) \\ (x_{26}, y_{26}, \theta_{26}) & = (2.9, -1.7, \frac{9}{10}\pi) \end{cases}$$

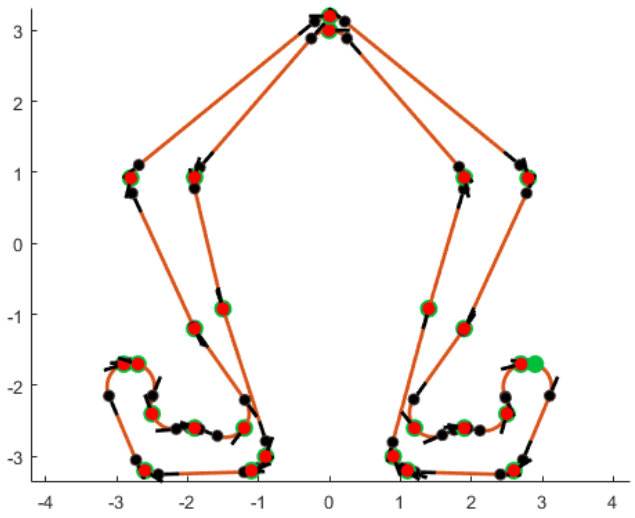


Figura A.5: Cammino ottimo con lunghezza 41.0725

A.6 Problema 6

Dimensione Problema: N = 6
Raggio di curvatura: $r = 1$

$$\begin{cases} (x_0, y_0, \theta_0) & = (0, 0, \frac{\pi}{2}) \\ (x_1, y_1) & = (2, 10) \\ (x_2, y_2) & = (4, 0) \\ (x_3, y_3) & = (6, 10) \\ (x_4, y_4) & = (8, 0) \\ (x_5, y_5) & = (10, 10) \\ (x_6, y_6, \theta_6) & = (12, 0, -\frac{\pi}{2}) \end{cases}$$

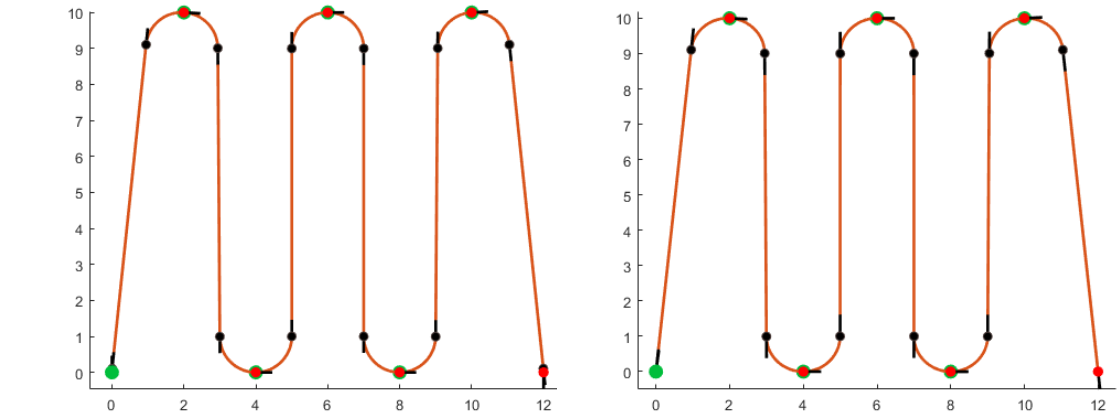


Figura A.6: A sinistra il cammino ottimo del tipo RSRSLSRSLSRSR con lunghezza 65.8132 mentre a destra il cammino ottimo con le direzioni di partenza e arrivo libere del tipo SRSLSRSLSRS con lunghezza 65.8129

A.7 Problema 7

Dimensione Problema: N = 3
Raggio di curvatura: $r = 1$

$$\begin{cases} (x_0, y_0, \theta_0) & = (0, 0, \frac{\pi}{2}) \\ (x_1, y_1) & = (1, 5) \\ (x_2, y_2) & = (5, -3) \\ (x_3, y_3, \theta_3) & = (8, 0, 0) \end{cases}$$

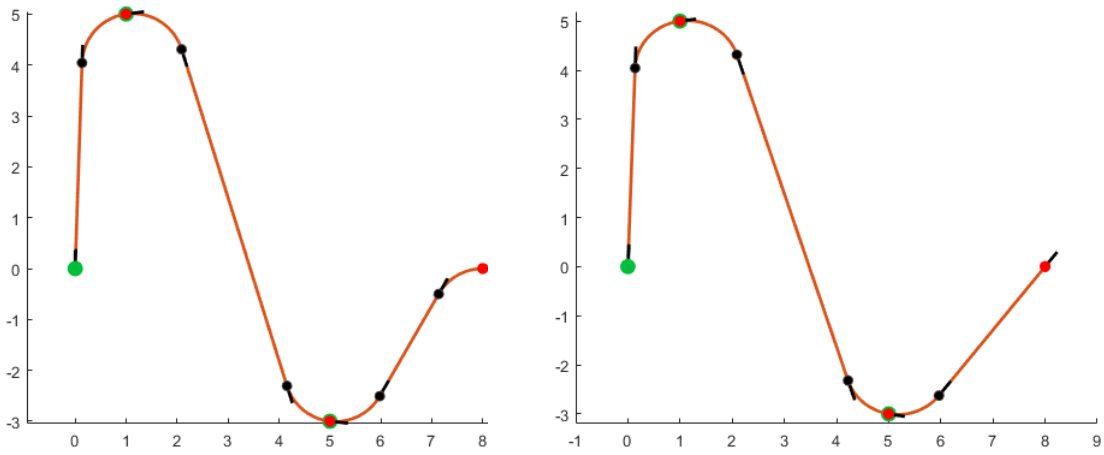


Figura A.7: A sinistra il cammino ottimo del tipo RSRSLSR con lunghezza 19.4558 mentre a destra il cammino ottimo con le direzioni di partenza e arrivo libere del tipo SRSLS con lunghezza 19.3087

Appendice B

Pseudocodice

In questa appendice è possibile trovare gli pseudocodici dei programmi utilizzati nel corso della tesi.

B.1 Algoritmo di Frego et al.

Il seguente pseudocodice è stato implementato in Matlab nella versione R2021a facendo uso del Navigation Toolbox. Il codice è facilmente adattabile ai casi in cui le direzioni di partenza e/o arrivo non sono fissate.

Algoritmo di Frego et al. per cammini di Markov-Dubins [10]

DATI: $u_0, ..., u_N, \theta_0, \theta_N$
INPUT: $K \geq 4$ (numero di discretizzazioni dell'angolo)
$nr \geq 1$ (numero di raffinamenti)
$r \geq 0$ (raggio di curvatura) ;

Tratto = dubinsConnection;
Tratto.MinTurningRadius $\leftarrow r$;
$h \leftarrow \frac{2\pi}{K}$
for j = 1 : K **do**
  $\theta_{N-1} \leftarrow (j-1) * h$
  [pathSegObj, Costo] = connect(Tratto,u_{N-1}, u_N);
  lunghezza(j) = Costo;
end for
for j = 1 : (N-2) **do**
  for m = 1 : K **do**
    for n = 1 : K **do**
      $\theta_{N-j-1} \leftarrow (m-1) * h$
      $\theta_{N-j} \leftarrow (n-1) * h$
      [pathSegObj, Costo] = connect(Tratto,u_{N-j-1}, u_{N-j});
      lunghezza2(m,n) = Costo;
    end for
  end for
  for m = 1 : K **do**
    for n = 1 : K **do**

```
        lunghezzaTOT(m,n) = lunghezza2(m,n)+lunghezza(n);
      end for
      [minval,posidx] = min(lunghezzaTOT(m,:));
      lungh(m) ← minval
      theta(m,N-j) ← (posidx-1)*h
    end for
    lunghezza ← lungh
end for
for j = 1 : K do
    θ_1 ← (j − 1) ∗ h
    [pathSegObj, Costo] = connect(Tratto,u_0, u_1);
    lunghezza3(j) = Costo;
end for
for j = 1 : K do
    lunghezzaFINALE(j) ← lunghezza3(j) + lunghezza(j)
end for
[minval,posidx]=min(lunghezzaFINALE);
Ottimo ← minval
θ_1^{0*} ← (posidx − 1) ∗ h
for j = 2 : (N-1) do
    θ_j^{0*} ← theta(θ_{j−1}^{0*}/h + 1, j)
end for
for i = 1 : nr do
    r_i ← (3/2) h_{i−1}
    h_i ← 2r_{i−1}/K
    for j = 1 : (K+1) do
        θ_{N−1} ← θ_{N−1}^{(i−1)*} − r_i(j − 1) ∗ h_i
        [pathSegObj, Costo] = connect(Tratto,u_{N−1}, u_N);
        lunghezza(j) = Costo;
    end for
    for j = 1 : (N-2) do
        for m = 1 : (K+1) do
            for n = 1 : (K+1) do
                θ_{N−j−1} ← θ_{N−j−1}^{(i−1)*} − r_i + (m − 1) ∗ h_i
                θ_{N−j} ← θ_{N−j}^{(i−1)*} − r_i + (n − 1) ∗ h_i
                [pathSegObj, Costo] = connect(Tratto,u_{N−j−1}, u_{N−j});
                lunghezza2(m,n) = Costo;
            end for
        end for
        for m = 1 : (K+1) do
            for n = 1 : (K+1) do
                lunghezzaTOT(m,n) = lunghezza2(m,n)+lunghezza(n);
            end for
            [minval,posidx] = min(lunghezzaTOT(m,:));
            lungh(m) ← minval
            theta(m,N-j) ← θ_{N−j}^{(i−1)*} − r + (posidx − 1) ∗ h_i
        end for
        lunghezza ← lungh
```

end for
for j = 1 : (K+1) **do**
$\theta_1 \leftarrow \theta_1^{(i-1)*} - r + (j-1) * h_i$
[pathSegObj, Costo] = connect(Tratto,u_0, u_1);
lunghezza3(j) = Costo;
end for
for j = 1 : (K+1) **do**
lunghezzaFINALE(j) ← lunghezza3(j) + lunghezza(j)
end for
[minval,posidx]=min(lunghezzaFINALE);
Ottimo ← minval
$\theta_1^{i*} \leftarrow \theta_1^{(i-1)*} - r + (posidx - 1) * h_i$
for j = 2 : (N-1) **do**
$\theta_j^{i*} \leftarrow theta((\theta_{j-1}^{i*} - \theta_{j-1}^{(i-1)*} + r_i)/h_i + 1, j)$
end for
end for

B.2 Algoritmo di Dhulipala et al.

DATI: $u_0, ..., u_N, \theta_0$
for $\forall S \subset 2^{u_1,...,u_N}$ **do**
$C(\{u_l\}, l) = c_{1l}, l \in \{u_1, ..., u_N\}$ per $|S| = 1$
$C(S, l) = \min_{m \in S - \{u_l\}} [C(S - \{u_l\}, m) + c_{ml}]$ per $|S| > 1$ dove c_{ml} è ottenuto risolvendo il problema di Dubins rilassato.
$\alpha(S, l)$ è la direzione in u_l ottenuta dopo essere partiti da u_0, aver visitato tutti i punti in S e aver concluso in $u_l \in S$.
end for
$\mathcal{C} = \min_{l \in \{u_1,...,u_N\}} [C(u_1, ..., u_N, l) + c_{l0}]$
$\alpha(\{u_1, ...u_N\}, u_0)$ è l'angolo di arrivo in u_0 dopo aver concluso il tour
Trova i_N tale che $\mathcal{C} = C(\{u_1, ..., u_N\}, i_N) + c_{i_n 0}$
$\alpha_{N+1} = \alpha(\{u_1, ...u_N\}, u_0)$
for $k = N - 1 : 1$ **do**
Trova i_k tale che $C(\{i_1, ..., i_{k+1}\}, i_{k+1}) = C(\{i_1, ..., i_k\}, i_k) + c_{i_k i_{k+1}}$
$\alpha_{k+1} = \alpha(\{i_1, ...i_{k+1}\}, i_{k+1})$
end for
return La sequenza $u_0, i_1, ..., i_N, u_0$ con i corrispondenti angoli di passaggio $\theta_0, \alpha_1, ..., \alpha_{N+1}$

Bibliografia

[1] Alessandro Agnoli. *Controllo di veicoli anolonomi su ruota*. PhD thesis, Tesi di laurea Specialistica, Università degli studi di Padova, 2007.

[2] Mohanad Al Nuaimi. Analysis and Comparison of Clothoid and Dubins Algorithms for UAV Trajectory Generation. *Ph. D. Thesis*, 2014.

[3] Haim Kaplan Andrew V. Goldberg, Chris Harrelson and Renato F. Werneck. Efficient point-to-point shortest path algorithms. `https://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf`. Available online, retrieved 12 March 2022.

[4] Luitpold Babel. New heuristic algorithms for the Dubins traveling salesman problem. *Journal of Heuristics*, pages 1–28, 2020.

[5] Xuân-Nam Bui, J-D Boissonnat, Philippe Soueres, and J-P Laumond. Shortest path synthesis for Dubins non-holonomic robot. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2–7. IEEE, 1994.

[6] Xuân-Nam Bui and Jean-Daniel Boissonnat. *Accessibility region for a car that only moves forwards along optimal paths*. PhD thesis, INRIA, 1994.

[7] Li Dai and Zheng Xie. On the length of Dubins path with any initial and terminal configurations. *Pure and Applied Mathematics Journal*, 4(6):248–254, 2015.

[8] Eswara Venkata Kumar Dhulipala. Angle Bisector Algorithm and Modified Dynamic Programming Algorithm for Dubins Traveling Salesman Problem. *https://doi.org/10.36227/techrxiv.14767593.v1 Preprint*, 2021. Available online, retrieved 12 March 2022.

[9] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[10] Marco Frego, Paolo Bevilacqua, Enrico Saccon, Luigi Palopoli, and Daniele Fontanelli. An iterative dynamic programming approach to the multipoint Markov-Dubins problem. *IEEE Robotics and Automation Letters*, 5(2):2483–2490, 2020.

[11] Andy Giese. A comprehensive, step-by-step tutorial on computing Dubin's curves, 2014.

[12] Xavier Goaoc, Hyo-Sil Kim, and Sylvain Lazard. Bounded-curvature shortest paths through a sequence of points using convex optimization. *SIAM Journal on Computing*, 42(2):662–684, 2013.

[13] L. M. Graves. *The Theory of Functions of Real Variables.* New York and London, 1946.

[14] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[15] Gregory Hartman. The arc length parameter and curvature. `https://sites.und.edu/timothy.prescott/apex/web/apex.Ch12.S5.html`. Available online, retrieved 12 March 2022.

[16] Sikha Hota and Debasish Ghose. Optimal geometrical path in 3d with curvature constraint. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 113–118. IEEE, 2010.

[17] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[18] C Yalçın Kaya. Markov-Dubins interpolating curves. *Computational Optimization and Applications*, 73(2):647–677, 2019.

[19] Artelys Knitro. Nonlinear optimization solver. `https://www.artelys.com/knitro`.

[20] Yucong Lin and Srikanth Saripalli. Path planning using 3d dubins curve for unmanned aerial vehicles. In *2014 international conference on unmanned aircraft systems (ICUAS)*, pages 296–304. IEEE, 2014.

[21] Satyanarayana G Manyam and Sivakumar Rathinam. On tightly bounding the Dubins traveling salesman's optimum. *Journal of Dynamic Systems, Measurement, and Control*, 140(7):071013, 2018.

[22] André César Medeiros and Sebastián Urrutia. Discrete optimization methods to determine trajectories for Dubins' vehicles. *Electronic Notes in Discrete Mathematics*, 36:17–24, 2010.

[23] Jerome Ny, Eric Feron, and Emilio Frazzoli. On the Dubins traveling salesman problem. *IEEE Transactions on Automatic Control*, 57(1):265–270, 2011.

[24] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458, 2018.

[25] Christos H Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical computer science*, 4(3):237–244, 1977.

[26] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, 2017.

[27] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem with neighborhoods. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1555–1562. IEEE, 2017.

[28] Ira Pohl. Bi-directional search. *Machine intelligence*, 6:127–140, 1971.

[29] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[30] Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006.

[31] Ketan Savla, Emilio Frazzoli, and Francesco Bullo. Traveling salesperson problems for the Dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391, 2008.

[32] Ketan D Savla. *Multi UAV systems with motion and communication constraints*. University of California, Santa Barbara, 2007.

[33] Madhavan Shanmugavel, Antonios Tsourdos, Brian White, and Rafał Żbikowski. Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs. *Control engineering practice*, 18(9):1084–1092, 2010.

[34] Andrei M Shkel and Vladimir Lumelsky. Classification of the Dubins set. *Robotics and Autonomous Systems*, 34(4):179–202, 2001.

[35] Rostislav Staněk, Peter Greistorfer, Klaus Ladner, and Ulrich Pferschy. Geometric and lp-based heuristics for the quadratic travelling salesman problem. *arXiv preprint arXiv:1803.03681*, 2018.

[36] Kaarthik Sundar, Sujeevraja Sanjeevi, and Christopher Montez. A branch-and-price algorithm for a team orienteering problem with fixed-wing drones. *EURO Journal on Transportation and Logistics*, page 100070, 2022.

[37] Héctor J Sussmann. Shortest 3-dimensional paths with a prescribed curvature bound. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 4, pages 3306–3312. IEEE, 1995.

[38] R. Rivest T. Cormen, C. Leiserson and C. Stein. Introduction to algorithms, 3rd ed. In *pag. 364*. Cambridge, MA: MIT Press, 2001.

[39] Amila Thibbotuwawa, Grzegorz Bocewicz, Peter Nielsen, and Zbigniew Banaszak. Unmanned aerial vehicle routing problems: a literature review. *Applied sciences*, 10(13):4504, 2020.

[40] Petr Váňa and Jan Faigl. Optimal solution of the Generalized Dubins Interval Problem: finding the shortest curvature-constrained path through a set of regions. *Autonomous Robots*, 44(7):1359–1376, 2020.

[41] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[42] Xiaoliang Wang, Peng Jiang, Deshi Li, and Tao Sun. Curvature continuous and bounded path planning for fixed-wing uavs. *Sensors*, 17(9):2155, 2017.

[43] Dongxiao Yang, Didong Li, and Huafei Sun. 2D Dubins path in environments with obstacle. *Mathematical Problems in Engineering*, 2013, 2013.