Sulla ricerca del cammino minimo con ostacoli poligonali convessi disgiunti su piano

Niccolò Della Rocca

***Abstract*—Questo testo descrive ed analizza dati sperimentali relativi ad algoritmi di ricerca applicati su una classe di problemi di rilevanza pratica nella guida autonoma di robots, oltre a descrivere proprietà di questa famiglia di problemi.**

I. Introduzione

Si consideri la seguente classe di problemi: sono assegnati un piano infinito, un punto di partenza S, un punto di destinazione G e si vuole trovare il cammino più breve tra i due. Ogni punto del piano, rappresentato tramite una coppia *(x,y)*, è uno stato e lo spazio degli stati indotto è continuo, infatti stabilita un'origine arbitraria nel piano e due assi cartesiani si può decidere di prendere ad esempio $S = (0, 0)$ e $G = (1, 0)$ e tra i due esistono infiniti punti, e quindi infiniti stati, anche nel cammino di lunghezza minima, cioè quello che li connette in linea d'aria. Anche i cammini tra S e G sono infiniti, infatti si possono considerare cammini lineari della forma $S \rightarrow M \rightarrow G$ dove M è un qualsiasi punto lungo l'asse del segmento $\overline{SG}$, e questi costituiscono una famiglia di cammini parametrizzati dall'angolo con segno $S\hat{M}G$ che è un parametro reale continuo. Reintroducendo la presenza di un insieme di poligoni convessi che fungono da ostacoli il numero di cammini varia a seconda della loro disposizione sulla scena ed al limite potrebbero non esistere cammini tra i due punti selezionati, se tutti quelli possibili in assenza di ostacoli sono adesso occlusi.

II. Proprietà della soluzione

In questa sezione discutiamo informalmente una proprietà del cammino di lunghezza minima nel contesto descritto nella sezione precedente. Tale proprietà è espressa dal seguente risultato:

Proposizione II.1. *Assegnato un insieme di ostacoli poligonali disgiunti su un piano e due punti S e G su di esso, il cammino minimo tra questi consiste in una linea spezzata.*

A sostegno di quanto proposto muoviamo la seguente argomentazione: assumiamo per assurdo che in realtà il cammino Γ di lunghezza minima tra S e G sia curvo. In questo caso esiste almeno una circonferenza contenuta nello spazio vuoto che contiene una sezione curva del cammino minimo, e quest'ultima può essere sostituita col segmento che unisce i due punti in cui la curva interseca la circonferenza. Il cammino così ottenuto è localmente più corto rispetto a Γ e questo contraddice l'ipotesi che Γ sia il cammino minimo, infatti per essere tale deve essere il più breve in ogni sua sottosezione, cioè localmente. Una intuizione del procedimento applicato è illustrata dalla Figura 1.

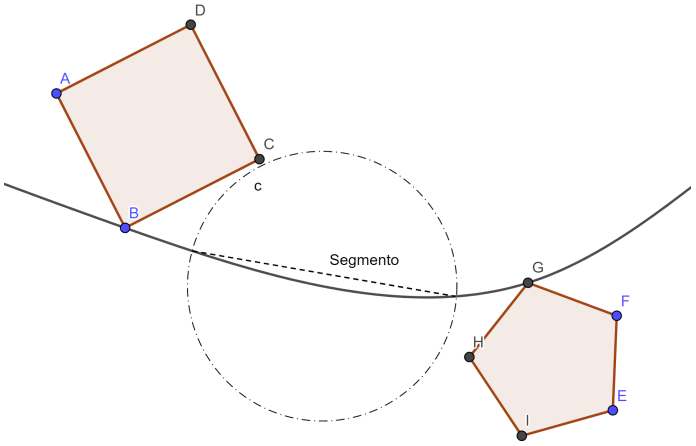

Fig. 1: Sostituzione sezione curva con segmento, prop. II.1

Una volta convinti della correttezza della proposizione II.1 è intuibile che vale anche quanto segue:

Proposizione II.2. *Assegnato un insieme di ostacoli poligonali disgiunti su un piano e due punti S e G su di esso, il cammino minimo tra questi è una linea spezzata che unisce una sequenza di vertici di poligoni, ad eccezione al più dei punti estremi.*

Non dimostriamo formalmente il risultato; l'intuizione che ci guida in tal senso è che in assenza di ostacoli, oppure se un cammino potesse attraversare questi, quello minimo come già evidenziato sarebbe quello in linea d'aria tra S e G. Il fatto che gli ostacoli siano solidi ci impone di aggirarli e la proposizione II.1 stabilisce che occorre farlo usando linee spezzate. Una qualsiasi linea spezzata che non congiunge due vertici di poligoni può essere sostituita con una più corta avente per estremi due vertici da cui segue II.2.

III. Definizione del problema di ricerca

I risultati ottenuti fanno uso della formulazione del problema di ricerca descritta in questa sottosezione. Lo spazio degli stati è formato da ogni possibile vettore spostamento da un vertice o punto di partenza o arrivo ad un altro. Lo stato iniziale è il vettore nullo applicato nel punto di partenza ed uno stato è un goal se il suo secondo estremo è il punto di destinazione. Un'azione applicabile in uno stato $\vec{s} = (s_0, s_1)$ è uno qualsiasi dei punti nella scena visibili da s_1, ed il risultato di una di queste sarà di cambiare lo stato da $\vec{s}$

a $\vec{s}'(s_1, P)$ con P uno dei punti visibili in s_1. Il costo di un'azione è definito come la norma euclidea del vettore spostamento risultante dall'applicazione dell'azione stessa, od in altri termini $d_2(s_1, P)$.

IV. Costruzione di una scena

In questo testo chiamiamo *scena* una qualsiasi istanza del problema trattato, cioè un piano con un insieme di ostacoli poligonali convessi disgiunti con un punto di partenza ed uno di arrivo, che possono corrispondere a vertici oppure essere punti aggiuntivi. Il programma realizzato per analizzare gli algoritmi è dotato di una funzionalità che consente la generazione di una scena "casuale", ed opera nel modo seguente.

1) Dato un parametro $r_{min} \geq 21$ viene generato un insieme massimale di campioni tramite un processo di *Poisson disk sampling* distanziati l'uno dall'altro almeno di r_{min}, che sono salvati in un array.
2) L'array di campioni viene permutato in maniera casuale e vengono estratti da essi i primi p elementi, con p un parametro impostato esternamente che rappresenta il numero di poligoni da generare.
3) Per ciascuno punto estratto p_i si calcola la minima distanza d_i dai rimanenti con l'ausilio di un 2D-tree.
4) Intorno a ciascun punto si costruisce un cerchio con raggio $21 \leq r < d_i/2$ generato secondo una distribuzione che privilegia raggi più vicini a $d_i/2$.
5) Per ciascun cerchio così costruito si generano v angoli casuali la cui somma sia 2π con $v \leq v_{max}$, parametro impostato esternamente. Questi angoli corrisponderanno ai vertici del poligono generato.
6) Si sceglie un poligono a caso ed un vertice a caso su di esso come punto di partenza, e si ripete un'altra volta per produrre un punto di destinazione.

Il campionamento tramite Poisson disk-sampling garantisce una distribuzione approssimativamente uniforme[1] dei circocentri dei poligoni generati, e costruire circonferenze di raggio massimo $d_i/2$ garantisce che tutti i poligoni generati saranno disgiunti.

Come suggerito al punto (4) si vogliono privilegiare i raggi più esterni facendo sì che i poligoni generati siano tendenzialmente grandi e costituiscano quindi ostacoli significativi. Per raggiungere questo obiettivo si usa una versione traslata e scalata della distribuzione esponenziale, tale che $\mathbb{P}(r_{min} \leq X < d_i/2) = 1$. Imponendo tale condizione si trova che la densità sarà siffatta:

$$\rho_i(r) = \begin{cases} 0 & \text{Se } r < r_{min} \\ \dfrac{\lambda}{e^{-\lambda r_{min}} - e^{-\lambda d_i/2}} \cdot e^{-\lambda t} & \text{Se } r_{min} \leq r < d_i/2 \\ 0 & \text{Se } r \geq d_i/2 \end{cases}$$

Il parametro λ è scelto di modo che almeno il 75% della distribuzione cada tra in nella prima metà di $[r_{min}, d_i/2)$. Vista la forma particolare della densità $\rho_i(r)$ si richiede piuttosto che $\mathbb{P}\left(X \leq 0.5(d_i/2 - r_{min})\right) \geq 0.75$ dove $\mathbb{P}_X \sim \text{Exp}(\lambda)$, il ché impone il seguente lower bound:

$$\lambda \geq \frac{1}{0.5\left(d_i/2 - r_{min}\right)} \cdot \ln\left(\frac{1}{1 - 0.75}\right)$$

Un valore x viene campionato dalla distribuzione in questione tramite la tecnica dell'inversione[2] ed il raggio che gli corrisponde sarà $d_i/2 + r_{min} - x$, in modo da correggere il fatto che la distribuzione prodotta prende valore nella prima metà di $[r_{min}, d_i/2)$ con probabilità di almeno 0.75 e non nella seconda, come invece si vorrebbe. Vengono inoltre selezionati casualmente due poligoni distinti, e su ciascuno di essi un vertice a caso; i punti così selezionati corrisponderanno ad un punto di partenza e di arrivo casuali. Non è quindi previsto che vengano generati punti estremi negli spazi vuoti da ostacoli in maniera casuale.

Infine, scegliere i vertici del poligono in maniera casuale lungo le circonferenze generate garantisce che tutti i poligoni prodotti saranno convessi.

V. Tecnica di sperimentazione

Nel corso della sperimentazione sono stati raccolti i dati relativi all'esecuzione di Uniform-Cost ed A* con due diverse euristiche, ossia d_1 e d_6, rispettivamente la distanza euclidea e distanza-6 dello stato dalla destinazione, con un numero variabile di poligoni nella scena. Sono stati testati vari insiemi di ostacoli con cardinalità tra 2 e 100, con passo di 2 per cardinalità comprese tra 2 e 20, e passo di 25 per insiemi più grandi. Per ciascuna cardinalità sono state generate 25 scene casuali con la tecnica descritta nella sezione precedente, con v_{max} generato casualmente tra 8 e 20 e diverso da scena in scena. Per ogni mappa così prodotta sono stati eseguiti gli algoritmi sopra menzionati e per ciascuno registrate informazioni rilevanti di cui dimensione di frontiera ed insieme degli esplorati, tempo di esecuzione, memoria ed Effective Branching Factor su un file CSV, poi elabolato tramite R.

Data la tendenza da parte del generatore casuale di scene, osservata durante la sperimentazione, di generare punti di partenza o di arrivo con assenza di ostacoli o comunque pochi di questi nel mezzo, per un numero di poligoni sufficientemente grande sono talvolta stati scelti manualmente dei punti estremi diversi da quelli generati automaticamente, nel tentativo di riequilibrare le "difficoltà" dei problemi prodotti.

Con i risultati prodotti dalla sperimentazione sono stati formati raggruppamenti per algoritmo e numero di poligoni nella scena, e per ciascuno di questi, formato da 25 entrate, si è calcolata la media dei parametri registrati.

VI. Commento prestazioni

I test sono stati realizzati su un PC che monta una CPU AMD Ryzen 7 5800X @ 3.8GHz, 16GB di memoria RAM @ 3600MHz, CL16, ed una GPU NVidia RTX 3070 FE.

I risultati sperimentali sono riassunti nella tabella sottostante. Da questa si evince che indipendentemente dalla dimensione del problema, riportata in termini di poligoni sulla scena, Uniform-Cost è sempre l'algoritmo che presenta

Costo ricerca												
	Nodi generati			Tempo (ms)			Memoria (kB)			Effective Branching Factor		
Poligoni	UC	A*(d_2)	A*(d_6)	UC	A*(d_2)	A*(d_6)	UC	A*(d_2)	A*(d_6)	UC	A*(d_2)	A*(d_6)
2	67.04	27.64	38.60	20.96	8.12	10.60	8.74	3.28	4.63	3.86	3.86	2.84
4	225.64	79.08	116.12	70.28	18.24	27.00	30.13	7.69	11.91	4.31	2.76	3.42
6	436.85	144.04	246.96	130.96	29.85	54.08	55.07	23.61	12.70	3.55	2.54	2.90
8	671.56	192.44	297.12	196.84	37.88	64.76	80.20	16.73	28.33	5.34	3.42	3.94
10	1105.76	223.60	389.88	348.92	41.72	76.68	124.08	17.62	32.85	5.45	3.20	3.91
12	1721.88	335.64	580.08	593.36	62.16	108.12	185.57	25.03	44.80	5.46	3.21	3.85
14	1918.44	450.68	696.44	686.16	91.68	145.64	214.70	36.93	58.17	5.32	3.22	3.92
16	2238.08	426.15	829.16	995.92	85.23	176.88	265.74	32.90	67.74	4.62	2.91	3.51
18	1979.56	342.84	576.20	823.68	67.04	115.88	232.94	26.34	45.38	4.88	3.04	3.43
20	2421.12	331.84	617.24	1296.40	72.08	135.04	301.14	26.00	49.44	4.77	2.87	3.34
25	3696.16	603.83	1171.92	2018.08	134.71	279.04	414.89	44.96	91.34	4.53	2.82	3.26
50	13275.04	1511.88	3044.58	18022.62	400.73	886.00	1554.95	106.2	222.82	6.99	3.87	4.75
75	25873.64	2799.44	7572.40	77025.40	957.12	3044.12	3126.54	191.32	550.81	5.85	3.37	4.25
100	29698.52	2381.40	6543.20	103363.68	945.84	3116.84	3724.73	162.95	489.98	8.34	3.52	4.69

Fig. 2: Tabella riassuntiva costi ricerca per varie dimensioni della scena

prestazioni peggiori sia in termini di costo della ricerca (nodi generati, tempo di esecuzione e memoria) che di EBF, a cui seguono nell'ordine A*(d_6) ed A*(d_2); tale gerarchia è mantenuta consistentemente per qualunque numero di poligoni. In particolare si osserva che il numero di nodi generati, il tempo di esecuzione e la memoria utilizzata tendono a crescere molto velocemente in UC rispetto ad A*, con un andamento che è per tutti questi parametri esponenziale nel numero di poligoni, cfr. Figura 4. Sempre in termini del tempo di esecuzione si osserva un consistente aumento della variabilità al crescere del numero dei poligoni: ad esempio con UC per 100 di questi nella scena si è registrato un tempo massimo di $457167ms$ (oltre 7 minuti) e minimo di $5955ms$ (poco meno di $6s$) per una media di $103363ms$ (poco meno di due minuti).

Per quanto riguarda l'Effective Branching Factor, questo sembra debolmente dipendente dalla dimensione del problema per tutti gli algoritmi, e presenta una bassa variabilità rispetto al valor medio ($\sigma_{\text{EBF-UC}} \simeq 1.24$, $\sigma_{\text{EBF-A6}} \simeq 0.59$ e $\sigma_{\text{EBF-A2}} \simeq 0.41$).

Circa il confronto tra i costi delle ricerche con A* ed euristiche diverse, si ricordi che detta $d_p(\cdot,\cdot)\colon \mathbb{R}^2 \times \mathbb{R}^2 \mapsto \mathbb{R}^+$ la distanza-p questa è monotona decrescente rispetto a p[3], pertanto d_2 forma un'euristica più informata rispetto a quella che adotta d_6. Ci si aspetta pertanto che con h_2 A* espanda un numero di nodi minore rispetto ad A* con h_6, ed i dati confermano la previsione: in media con la seconda vengono generati circa l'85.95% di nodi in più, con picchi intorno al 170% per dimensioni del problema più grandi, con conseguenti ripercussioni percettibili sia in termini di tempo di esecuzione che di memoria impiegata. L'effetto principale nell'aumento in percentuale dei nodi generati è dovuto soprattutto all'insieme degli esplorati, che per problemi grandi arriva a contenere oltre cinque volte tanto rispetto al caso in cui si impieghi h_2, mentre l'incremento percentuale della frontiera non supera il 150%.

Analizziamo adesso in maniera dettagliata un caso specifico, che è quello in cui si sono generate scene composte da 25 poligoni ciascuna. Come si evince dalla Figura 3 UNIFORM-COST, in linea con la tendenza generale, ha prestazioni

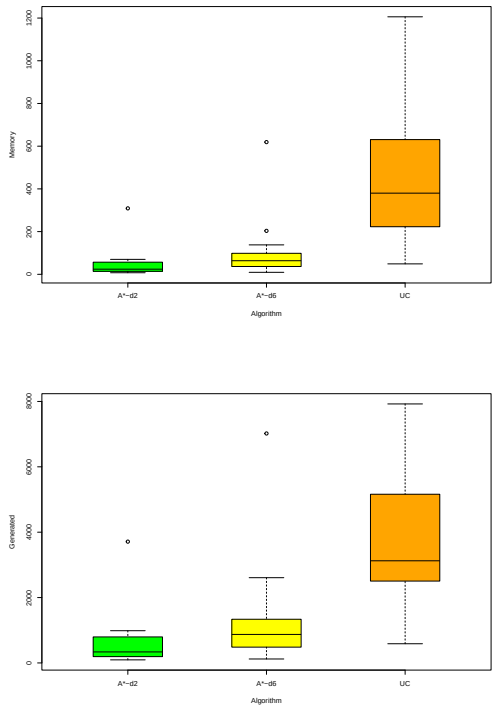

Fig. 3: Box plot per memoria (alto, in kB) e nodi generati (basso) nel caso di scene composte da 25 poligoni ciascuna.

notevolmente peggiori rispetto alle due varianti di A* sia per quanto riguarda il tempo di esecuzione, memoria richiesta e nodi generati, a tal punto che per tutti questi indici di prestazioni il primo quartile di UC (primo 25% della distribuzione) è vicino o addirittura inferiore al valore peggiore dei due A*.

La memoria massima registrata da UC è di circa 1.18MB mentre quella minima è di 48.87kB, la massima per A*(d_6) è di 619.09kB mentre la minima di 9.28kB, la massima per A*(d_2) è stata di 308.81kB e la minima di 7.36kB. Il 75%

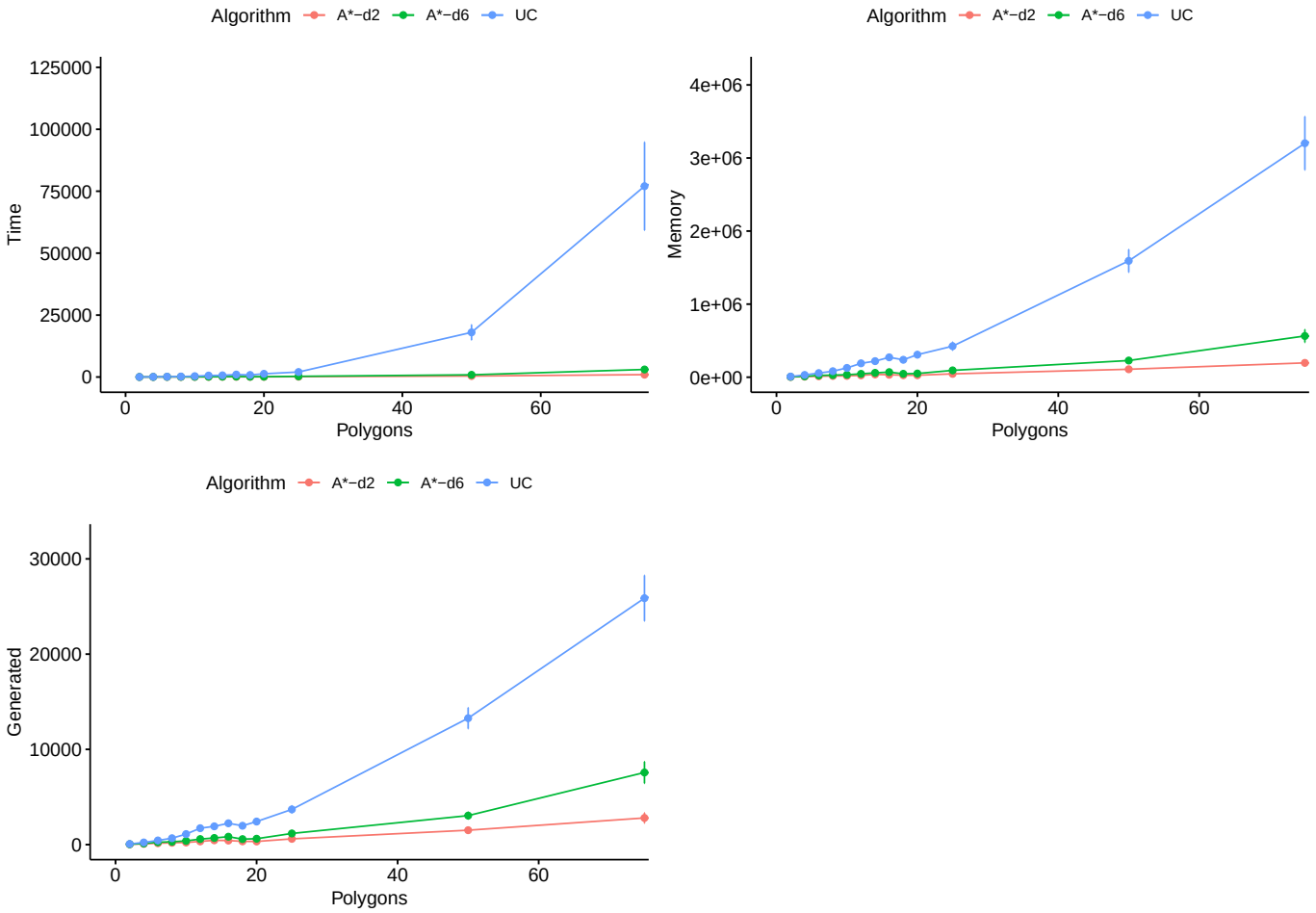

Fig. 4: Tre grafici che riportano rispettivamente l'andamento del tempo di esecuzione medio, utilizzo di memoria medio e numero medio di nodi generati in funzione delle dimensioni del problema.

dei valori di memoria registrati per A*(d_2) è inferiore al 50% di quelli rilevati per lo stesso algoritmo con d_6, e si osserva che la mediana per A*(d_2) è molto vicina al quartile di livello 0.25 suggerendo una bassa varianza per questo parametro.

Similmente per quanto riguarda la distribuzione del numero di nodi generati dai vari algoritmi si osserva che UNIFORM-COST presenta un'elevata variabilità ed un quartile di livello 0.25, pari a 2504 nodi, inferiore rispetto a quello di livello 0.75, corrispondente a 1298 nodi, di A* con d_6. Allo stesso modo, quando si usa d_2 come euristica si ottiene un quartile di livello 0.75, corrispondente a 794 nodi, inferiore alla mediana di A*(h_6), ossia 872 nodi.

La stessa gerarchia di prestazioni che emerge dall'analisi di questi due parametri, che vede A* con d_2 prevalere su tutta la linea sull'analogo con d_6, a sua volta decisamente migliore di UNIFORM-COST, si ripresenta in maniera ancora più pronunciata sui tempi di esecuzione, che vedono assegnare il primato di tempo peggiore a quest'ultimo, con ben 11.14s, contro 2.14s ed 1.06s di A*(d_6) e A*(d_2) rispettivamente. Per quanto riguarda il confronto da gli ultimi due, per questi la differenza è meno netta rispetto a quanto non lo fosse per gli altri parametri, con quartili 0.25-0.50-0.75 di 101ms-185ms-0.275ms per la variante con d_6 e 37ms-69ms-154ms per quella con d_2.

REFERENCES

[1] P.-T. B. P. V. Bhavya Kailkhura, Jayaraman J. Thiagarajan, "Theoretical guarantees for poisson disk sampling using pair correlation function," 2016.

[2] L. Devroye, *Non-Uniform Random Variate Generation, p28*. New York: Springer-Verlag, 1986.

[3] I. H. J. Mustapha Raissouli, "Various proofs for the decrease monotonicity of the schatten's power norm, various families of Rn norms and some open problems," 2010.