UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

"Valutazione di funzioni utilità per agenti distribuiti in sistemi veicolari a guida autonoma"

Relatore: Leonardo Badia

Laureando: Mattia Piana

ANNO ACCADEMICO 2020 – 2021

Data di laurea 23-09-2021

Indice

Abstract

Ai giorni nostri, con l'aumento costante sul pianeta della popolazione, il supporto della tecnologia nella vita quotidiana si sta rivelando sempre più cruciale, [15]. Alla pari dell'efficienza energetica, che in questi anni ha fatto da padrone, la progettazione urbanistica intelligente si colloca tra le maggiori sfide del nostro tempo. Lo scopo dell'elaborato si colloca in questo contesto, in particolare modellerò e studierò varie tipologie di scelta per la ricerca di un parcheggio per veicoli a guida autonoma in un Parcheggio Multi-Livello.

Introduzione

Uno dei paesi in cui l'aumento della popolazione, in particolare in contesto urbano, è una realtà quotidiana è la Cina. Basti pensare che dal 1978 al 2012, il tasso di urbanizzazione cinese è passato dal 17.92% al 52.57%: è stato stimato infatti che la popolazione che vive in città è aumentata di 10 milioni all'anno [1]. Sempre in [1], sono stati evidenziati gli effetti che ciò comporta sia in termini spaziali, che energetici: l'area urbana in Cina nel 2010 è quintuplicata rispetto al 1980 mentre i consumi in termini energetici sono quadruplicati. Un ulteriore effetto di questa rapida crescita, è rappresentata dal numero di veicoli in circolazione, ad esempio nella città di Beijing a fronte dei 5 milioni di veicoli presenti nel 2016 erano disponibili solamente 2 milioni di posteggi [6]. Va da sé che un aumento così rapido della popolazione, con associato un aumento vertiginoso dei veicoli in circolazione, richieda un'organizzazione efficiente.

Un'altra tematica che potrebbe beneficiare di un'organizzazione urbana efficiente è lo sviluppo di veicoli a guida autonoma, infatti una delle sfide aperte per la ricerca in questo ambito è data dall'interazione tra i diversi agenti e l'ambiente circostante [7]: essa avrebbe come effetto una diminuzione dei possibili scenari che un veicolo a guida autonoma sarebbe chiamato a gestire. I benefici di cui la società potrebbe godere della diffusione di veicoli intelligenti sono molteplici, basti pensare che in [5] viene riportato che la loro massiva introduzione porterebbe una riduzione dei consumi di carburante fino al 60% grazie alla gestione ottima dei veicoli e alla possibilità di comunicazione tra di loro.

Quello di cui mi sono occupato in questo elaborato è uno dei problemi riguardanti la progettazione urbanistica intelligente, ossia quello di trovare posteggio in un Parcheggio Multi-Livello. Per far comprendere l'importanza di questo problema, è stato stimato che tra il 1927 ed il 2001, nelle aree metropolitane come Detroit, Londra e New York, le auto che cercavano parcheggio costituivano tra l'8% e il 74% del traffico totale [8], da cui i relativi effetti sia sull'inquinamento che sul tempo perso nella ricerca. Per essere più precisi, in [14] viene riportato che in base ad uno studio effettuato in 11 città principali il 30% del traffico totale è dovuto a veicoli alla ricerca di un parcheggio. Infine, un sondaggio condotto in [16] ha permesso di constatare le problematiche percepite, ed è emerso che la problematica principale, a parimerito con la sicurezza, è il sovraffollamento della struttura.

Il mio contributo consiste nel definire delle strategie euristiche per la ricerca di parcheggio in un parcheggio multilivello, e valutarne l'efficacia. La valutazione avverrà mediante l'introduzione di una *Utility Function*, che tiene conto sia del tempo impiegato dai vari agenti per la ricerca che del tempo di uscita dal parcheggio. L'approccio che verrà adottato, data la complessità del problema, sarà la simulazione: utilizzerò un programma MatLab scritto da me per simulare vari scenari, cercando allo stesso tempo di verificare la correttezza del codice quando possibile.

Il lavoro si articola come segue: nella sezione I vengono presentati i principali studi presenti in questo ambito o affini, mentre nella sezione II viene presentato il modello che ho deciso di adottare per la simulazione. Nella sezione III vengono studiate le diverse politiche a priori, in cui ogni agente segue lo stesso pattern di ricerca. In questa sezione sono presenti anche gli effetti che il cambiamento dei parametri strutturali del parcheggio può apportare in termini di Utility. Nella sezione IV sono presenti le strategie probabilistiche, in cui l'ordine di visita viene deciso in maniera casuale da ciascun agente. Nella sezione V vengono studiate ulteriori strategie che, a differenza delle precedenti, sfruttano informazioni ulteriori sullo stato del parcheggio oppure sulle scelte già intraprese dagli agenti che hanno visitato la struttura in precedenza. Nella sezione VI sono presenti le conclusioni dello studio, mentre nell'*Appendice* sono presenti alcuni commenti sul codice del simulatore e sulle differenze tra il mio modello e quello adottato in [2].

Sezione I: Stato dell'arte

Sfortunatamente la letteratura scientifica è sorprendentemente carente in questo ambito. Spesso i diversi articoli si focalizzano sulle diverse tecnologie che si possono utilizzare, ad esempio Machine Learning in [12], da impiegare in scenari di parcheggi multilivello e auto a guida autonoma del futuro, oppure simulando l'impatto che la ricerca di parcheggio può avere sulla congestione del traffico, [9]. In [10] viene studiato come l'adottare un Intelligent Parking Guidance Systems (IPGS), ossia un sistema integrato di Telecomunicazioni e sensori, possa aiutare i diversi automobilisti nel trovare posteggio risparmiando sia tempo che emissioni rispetto un sistema di ricerca classico che si basa nel cercare nella struttura più vicina.

In [2] e [14] il problema di ricerca del parcheggio viene studiato come uno scenario di Game Theory in cui la competizione tra i veicoli nella ricerca di un posteggio viene interpretata come un gioco. Nella prossima sezione, definirò un modello simile a [2] ed analizzerò come differenti strategie euristiche possano migliorare o peggiorare il costo associato alla ricerca. Per fare ciò dovrò fissare dei vincoli di simulazione, ed è proprio questa una delle difficoltà nel creare un modello per quanto possibile fedele alla realtà.

Un primo problema che si incontra riguarda la caratterizzazione degli utenti e delle loro strategie. Nella realtà, le linee guida che governano il comportamento degli utenti/automobilisti possono essere innumerevoli, anche se può giovare tentare una classificazione secondo alcune macro-categorie. In [13] infatti abbiamo un'indagine del Transport Studies Unit in Kingston e Birmingham svolta su una platea di automobilisti inerente le loro "Strategie di parcheggio", ed in effetti le strategie che proporrò possono essere ricondotte alle categorie lì citate. Però in [13] il contesto è molto più generale, visto che vengono prese in considerazione tutte le tipologie di parcheggio (anche quelle non a norma di legge) ed inoltre queste strategie possono variare a seconda della struttura/conoscenza del parcheggio da parte degli automobilisti e dal loro orario di arrivo.

Nonostante non abbia trovato molti studi specifici sulla gestione di un Multy-Storey-Parking-Garage, in letteratura ne sono presenti diversi sull'arte della ricerca del parcheggio in generale. In [17] viene condotto uno studio su tre semplici strategie nel contesto di un parcheggio ad un solo livello, caratterizzato da un target (che può essere ad esempio la zona adibita ad un evento). L'obiettivo degli automobilisti è parcheggiare più vicino possibile al target, cercando di minimizzare il tempo di ricerca. Le strategie sono:

1- Meak Strategy: parcheggio nel primo posto libero (per ipotesi, i veicoli cominciano la ricerca dal posteggio più lontano possibile dal target).
2- Prudent Strategy: parcheggio nel posteggio più a sinistra del primo gap disponibile, ossia parcheggio dopo la prima vettura che trovo.
3- Optimistic Strategy: vado direttamente al target, ed in caso torno indietro fino a trovare il primo posto disponibile.

Dallo studio emerge che la migliore, in termini di tentativi e di vicinanza al target, è la Prudent Strategy, nonostante abbia come problematica la possibile non occupazione dei posti più vicini al target (cosa che, viene riportato in [17], può causare anche un problema di tipo "psicologico" ai guidatori, visto che andando poi verso il target vedrebbero dei posti liberi migliori rispetto ai loro). Questo risultato contrasta in realtà con [18], in cui la strategia migliore dovrebbe essere nell'andare più vicino possibile al target, così da sfruttare pienamente il sistema.

Infine, in [19] è presente uno studio interessante sulla possibilità di gestire il traffico attraverso differenti politiche di pricing. Dopo uno studio preliminare per identificare le aree e gli orari in cui il traffico è maggiore, quello che è stato fatto consiste nell'applicare costi diversi in diverse aree di parcheggio (sia quelle in strada che in parcheggi multilivello) ed orario. Dopo brevi periodi di prova sul campo per aggiustare il tiro, questa strategia si è rivelata vincente in diversi casi di studio sia in città molto grandi come San Francisco che città di minori dimensioni come Brescia. C'è da tenere presente che in [19] l'obiettivo era quello di tenere il livello di occupazione delle diverse strutture tra il 60% e l'80%, e non direttamente il tempo di ricerca (che comunque risente del livello di occupazione). Inoltre, per attuare questa tecnica, è necessario un costante monitoraggio delle diverse strutture. Nel seguente capitolo verranno discussi alcuni dei diversi approcci al problema presenti in letteratura, approfondendo alcuni di quelli già citati ed introducendone di nuovi.

Soluzioni attuali

In [11] vengono presentati i principali sistemi di parcheggio attualmente in uso in contesto urbano. A seconda dei servizi e dell'equipaggiamento di cui sono dotate le strutture, vengono classificate in 4 tipologie:

1- *Typical surface parking system*: è la soluzione più semplice, in cui non sono presenti particolari strumenti per guidare gli automobilisti e non vengono erogati servizi oltre alla necessaria manutenzione.
2- *Semi-automatic surface system*: per l'accesso alla struttura è necessaria la presenza di un operatore che alza le barriere di accesso, e ciò rappresenta anche il problema di questo sistema.
3- *Automatic surface system*: consiste in una struttura equipaggiata per operare in maniera completamente automatica. È dotata di sportelli automatici in cui è possibile acquistare il ticket di accesso oltre a strumenti di controllo come telecamere.
4- *Multi-storey car parks*: è la soluzione adottata quando non è possibile espandere l'area di parcheggio, come in centro città. Solitamente sono strutture automatiche e ben equipaggiate.

Per guidare gli automobilisti verso la struttura del parcheggio ci sono due tipologie di segnaletica: la segnaletica "offline" che consiste nei classici cartelli stradali e quella "online" in cui i cartelli dinamici forniscono informazioni aggiuntive sullo stato della struttura, come il numero di posti disponibili.

Offline *Online*

Dal punto di vista tecnologico sono presenti diversi metodi per aiutare gli automobilisti nella ricerca di un posteggio libero, che si differenziano per costi e servizio offerto. Una ricerca efficiente può riflettersi non solo in un risparmio di tempo, ma anche di emissioni (come si avrà modo di apprezzare nei lavori che seguono, vedi [10]). L'introduzione di queste tecnologie ha sicuramente un costo non indifferente, ma dal punto di vista amministrativo possono fornire informazioni/statistiche utili sul comportamento dei guidatori. Successivamente queste informazioni possono essere utilizzate per la gestione ottima del parcheggio, [19] e [21] ne sono un chiaro esempio. In [11] vengono infine brevemente riportati tre sistemi "smart" per la gestione di un parcheggio multilivello.

Il primo, denominato *GPP PGS2*, si basa sull'utilizzo di sensori ad ultrasuoni posti in ciascun posteggio e da dei led che segnalano se esso è occupato o meno. Questi sensori sono collegati a dei display che forniscono informazioni riguardanti il numero di posteggi liberi in quella zona, utili ai guidatori per orientarsi all'interno della struttura. Tuttavia, è possibile che questa continua trasmissione di dati tra i vari elementi possa causare un traffico eccessivo, [12].

GPP PGS2 System

Il *CrossPark Parking System* può essere visto come un'evoluzione del sistema precedente. È dotato di sensori nei differenti posteggi (ottici, anziché ad ultrasuoni) e fornisce anch'esso in tempo reale informazioni utili agli automobilisti, guidandoli nella ricerca del parcheggio. Al giorno d'oggi i sensori possono essere sostituiti da telecamere che monitorano 4 posteggi ciascuna, permettendo quindi un maggiore controllo dei vari utenti del sistema. È presente, inoltre, una funzionalità chiamata "Find my car" che, attraverso un identificativo univoco, permette di rintracciare il posteggio in cui si è lasciata l'auto e guidare l'utente verso di esso.

I seguenti due sistemi sono più futuristici e possono essere collocati in contesti di Smart Cities. Il primo, chiamato *Siemens Smart Parking System,* consiste nell'installazione di una rete di sensori ad infrarossi "on the road" nei parcheggi di tutta la città, a cui gli utenti possono interfacciarsi attraverso il proprio smartphone. Questi sensori, resistenti a temperature che vanno dai -10° ai +55°C, possono guidare gli automobilisti non solo verso un parcheggio libero, ma anche verso altre destinazioni. Inoltre, è possibile introdurre dei chip che permettano di identificare la tipologia di automobilista, dando così permessi di parcheggio speciali a persone che ne hanno diritto (ad esempio disabili). Un ultimo utilizzo dei sensori potrebbe essere quello di introdurre delle tariffe variabili nel corso della giornata, riducendo così il traffico ed emissioni. Potrebbe essere utile allo scenario di [19], in quanto velocizzerebbe il cambio di tariffa (anche se, viste le considerazione in [19], ciò potrebbe non essere applicabile dal punto di vista legale).

Sensore wireless on the road

L'ultimo sistema è l'*Advanced Parking Management System*, e prevede l'installazione di sensori nei lampioni e nelle facciate degli edifici che monitorano continuamente il livello di traffico in tutte le zone della città. I sensori emettono microonde direzionate in una certa area che, se incontrano un'autovettura, vengono riflesse. Così facendo viene rilevata la presenza di un'automobile. Successivamente i dati vengono raccolti dall'amministrazione cittadina, che li inoltra infine agli automobilisti. Il sistema è in grado anche di stimare se l'auto si trova in un parcheggio, fornendo così informazioni agli automobilisti sullo stato dello stesso; in caso fosse pieno, verranno fornite anche informazioni per come raggiungere quella particolare zona usando i mezzi pubblici.

Sensori multipli APMS

Il lavoro presentato in [12] consiste in un sistema intelligente per la gestione dei veicoli basato sull'IoT, e formato da tre parti principali:

1- *Parking Part*: è l'insieme di tutti i sensori presenti nella struttura, sia essa al chiuso oppure esterna, e dei microcontrollori dei dispositivi per la trasmissione dei dati.
2- *Cloud*: i dati raccolti dai sensori verranno poi trasferiti ad un cloud che fungerà da tramite tra i vari sistemi fisici e gli utenti.
3- *App Android*: gli utenti si interfacceranno al cloud mediante questa applicazione.

Il sistema funziona nella seguente maniera: dei semplici ed economici sensori IR vengono posti in ogni posteggio per rilevare se è libero o meno e, in tempo reale, inviano i dati via wireless ad un microcontrollore. Lo scopo di quest'ultimo è quello del filtraggio dei dati, infatti una delle problematiche emerse dal paper è la grande quantità di dati che il sistema è chiamato ad elaborare, e il fatto di adottare un'elaborazione distribuita è vitale per il funzionamento dello stesso. Il microcontrollore poi passa i dati al cloud ed esso ha un duplice compito: il primo è quello di applicare un algoritmo di Machine Learning ai dati in modo da salvarli in maniera efficiente e poi inviare le informazioni dello stato del parcheggio agli utenti. Purtroppo, non si scende in dettaglio sull'algoritmo da poter utilizzare, ed anzi viene proprio esplicitato che è una delle sfide aperte. Va da sé che la velocità di elaborazione da parte del cloud debba essere sufficientemente elevata per consentire il corretto funzionamento. L'utente infine comunicherà con il cloud attraverso internet mediante il protocollo http, ed esso fornirà all'utente informazioni real-time riguardanti il traffico e i posteggi liberi più vicini. Viene citata infine la funzionalità Text to Speech che permette agli utenti di non guardare il telefono durante la guida. Di seguito vengono riportate due immagini che schematizzano la struttura del sistema (*Fig.1*) e una possibile interfaccia dell'applicazione Android (*Fig.2*).

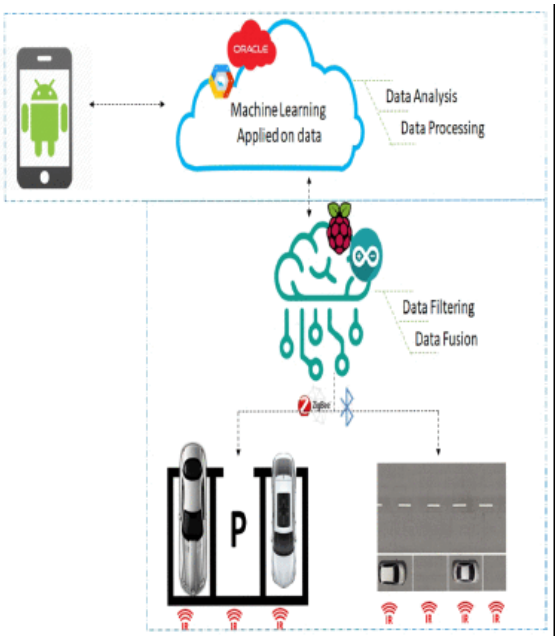


Fig.1

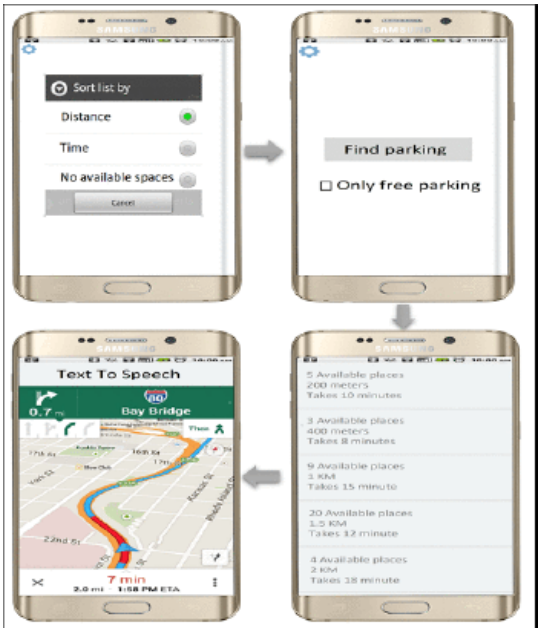
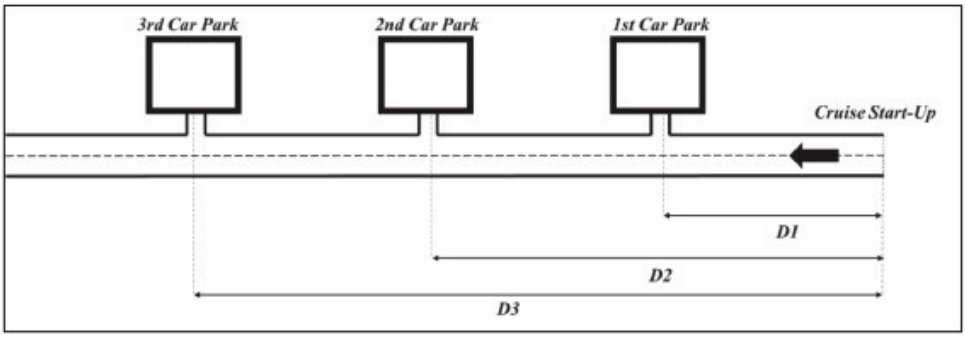


Fig.2

In [10] vengono confrontati il sistema IPGS ed i sistemi convenzionali Cs in termini di tasso di occupazione, tempo di ricerca e produzione di gas nocivi. L' IPGS può essere visto come una generalizzazione del sistema adottato in [12], quindi in maniera indiretta ne fornisce le motivazioni che possono portarne alla realizzazione a fronte di un investimento non indifferente. L'approccio che è stato adottato è di tipo simulativo, ed è stato utilizzato un programma open-source di nome NetLogo, messo a disposizione dal Centre for Connected Learning and Computer-Based Modelling (CCL) della Northwestern University. Lo scenario è quello di tre parcheggi, con un solo livello, ciascuno dei quali ha 275 posti auto. Le auto entrano tutte da un unico punto, e a seconda della politica adottata si sposteranno nel primo, secondo o terzo parcheggio a distanza D1=1km, D2=1.5km e D3=2km rispettivamente dal punto di partenza. Di seguito riporto un'immagine esplicativa dello scenario.



Gli agenti vengono classificati in due tipologie: quelli che adottano una strategia Cs, ossia tentano nella struttura più vicina, e quelli che sfruttano una strategia IPGS e quindi possiedono ulteriori informazioni sullo stato del parcheggio. In particolare, l'informazione aggiuntiva che possiedono viene sintetizzata dalla seguente formula:

$$R_{ij} = \frac{T_{ij}}{MTBA_j * f_j}$$

dove:

R_{ij} è il grado di disponibilità del parcheggio j

$MTBA_j$ è il tempo medio che intercorre tra due arrivi consecutivi nel parcheggio j

T_{ij} è il tempo totale che l'i-esimo veicolo impiega a raggiungere il parcheggio j

f_j è il numero di posti liberi nel parcheggio j

Ad ogni parcheggio viene associato un valore di R_{ij} dall'i-esimo veicolo, e confrontando i vari valori il veicolo sceglierà di conseguenza.

Per fornire un parametro quantitativo per valutare le emissioni viene utilizzato il modello di [4]. I gas che vengono presi in considerazione sono: $CO_2, CO, HC\ e\ NO$. Viene definito quindi il tasso di emissione TP_x $[g/s]$:

$$TP_x = A_x + B_x * v + C_x * v^3 + D_x * a * v$$

dove:

$x = CO_2\ oppure\ CO, HC, \dots$

$v =$ velocità del veicolo $[m/s]$

$a =$ accelerazione $[m/s^2]$

$A, B, C, D =$ parametri di calibrazione

In realtà per i gas $CO, HC\ e\ NO$ il tutto andrebbe moltiplicato per un'ulteriore costante, che dipende dal gas in questione. Questa costante rappresenterebbe la frazione di gas che non viene filtrata nella marmitta; tuttavia, per semplicità di notazione non la indico in quanto può essere inglobata nei parametri di calibrazione.

Le simulazioni che sono state condotte tengono conto di differenti situazioni, vengono variati infatti i processi di arrivi e partenze, il numero di veicoli in arrivo ed il numero di veicoli già presenti inizialmente nel sistema. Il primo processo consiste in arrivi random e senza partenze, mentre il secondo consiste in processi di arrivi e partenze di Poisson. Tuttavia, variando i processi di arrivi e partenze non ci sono variazioni significative nei risultati, in quanto i valori possono sicuramente cambiare, ma confrontando poi i risultati tra Cs e IPGS si traggono le stesse conclusioni.

I risultati riportati sono che l'adozione di un sistema IPGS porta ad una occupazione più omogenea dei diversi parcheggi, che si riflette in consumi minori. L'occupazione dei vari parcheggi raggiunge (nel sistema IPGS) un punto di equilibrio, che corrisponde ad un tasso di occupazione di circa l'80% (per lo scenario lì riportato) mentre la strategia Cs porta un livello di occupazione fortemente non uniforme, come c'era da aspettarsi. Nel caso un veicolo entri in un parcheggio già pieno, è stato scelto di applicare una penalità a tutti i veicoli: ciò avviene solamente nel caso nella simulazione di tipo Cs, infatti per nessun scenario (in cui vengono variati il numero di utenti) adottando un sistema IPGS si riempie completamente un parcheggio. Di seguito riporto due immagini che mostrano come varia il livello di occupazione dei parcheggi in funzione dei differenti scenari con processo di

Poisson (*Fig.1*). In (*Fig.2*) viene riportato il tasso medio di occupazione dei livelli, mentre in (*Tab.1*) è presente una tabella che riassume i vari scenari.

Scenario	Car Park (C.P.) Startup Occup.			Total arrival (veh.)
	1st C.P.	2nd C.P.	3rd C.P.	
1st	0	0	0	700
2nd	225	0	0	550
3rd	100	100	100	450
4th	175	150	125	250
5th	125	150	175	250

Tab.1

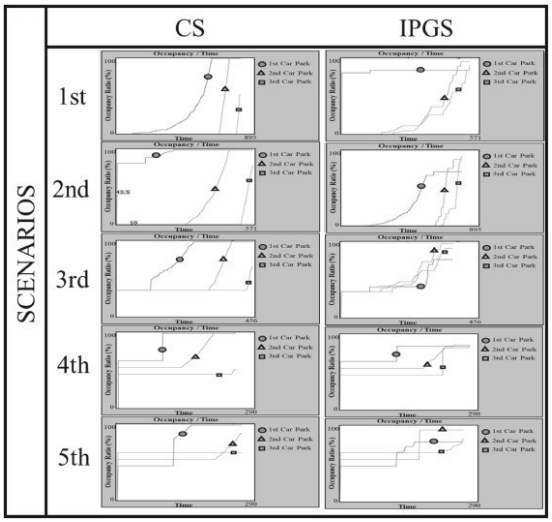


Fig.1

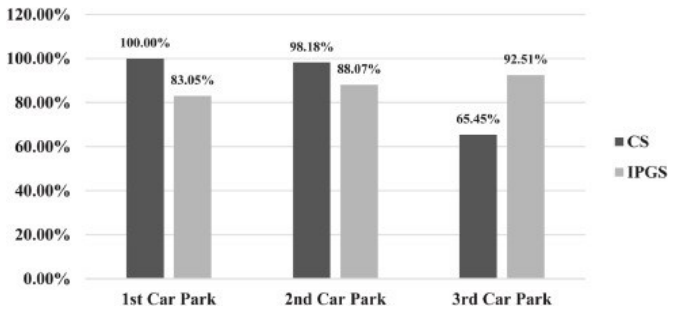
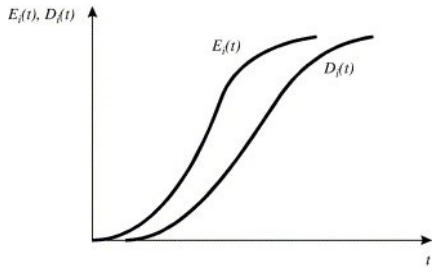


Fig.2

I sistemi di parcheggio "guidato" analizzati finora cercano in qualche modo di aiutare gli automobilisti mentre sono già arrivati nella struttura adibita al parcheggio. In [21] invece viene proposto un sistema che, combinando simulazioni, tecniche di ottimizzazione e *fuzzy rules*, permette di gestire in maniera efficiente il traffico. Questo sistema permette ai guidatori di trovare parcheggio anche **prima** di mettersi in viaggio, eventualmente arrivando a rifiutare alcune richieste, e può essere applicato in diversi contesti, dai parcheggi negli aeroporti a quelli multilivello in città.

Prima di addentrarsi nei dettagli del nuovo sistema, [21] di fatto conferma indirettamente i risultati di [19], infatti elenca alcuni benefici che una politica di pricing ad hoc ha apportato in diverse città come Madison (Wisconsin), Chicago e Seattle. In particolare, si rivelata molto utile per ridurre il livello di congestione del traffico (questa tecnica viene comunemente chiamata "Congestion-Pricing", appunto), inducendo gli automobilisti a spostarsi in certe zone piuttosto che altre o all'utilizzo di mezzi di trasporto alternativi negli orari di punta. Ciò implica che le tariffe possono variare nel corso della giornata e da veicolo a veicolo: son previste infatti tariffe maggiori per l'uso improprio dei parcheggi, ad esempio parcheggiare in una zona non adibita al proprio veicolo. Il sistema proposto prende spunto quindi dalle politiche di pricing, e tenta di risolvere il problema che segue:

"Dato un parcheggio in cui sono presenti m tariffe distinte $T_1, T_2, \dots, T_m$ e disponendo di sufficienti realizzazioni dei processi stocastici $E_i(t)$ e $D_i(t)$ (che sono rispettivamente i processi cumulativi di arrivi e partenze per la tariffa i-esima durante la giornata), sviluppare un modello che permetta di massimizzare i ricavi, avendo solamente la possibilità di accettare o meno la richiesta di parcheggio degli utenti"



Esempio di una possibile realizzazione dei processi $E_i(t)$ e $D_i(t)$

Il problema è più complesso di quanto sembra. Non essendo possibile accettare in generale tutte le richieste per il numero limitato di posti, sono necessarie delle scelte, tenendo in conto che spesso gli utenti prenotano un posteggio per poi non presentarsi oppure si presentano senza prenotazione. In letteratura vengono citate un paio di soluzioni: nella prima ad ogni zona è associata una tariffa con un numero di posteggi prestabilito. Il problema di questo metodo sono i posti inutilizzati (ad esempio si potrebbe essere costretti a rifiutare alcune richieste della tariffa più bassa tenendo dei posti liberi che poi non verranno sfruttati dalle altre tariffe, il che fa perdere denaro). La seconda soluzione prevede una suddivisione "annidata" dei posteggi, in cui la richiesta di parcheggio per una tariffa generica T_i non può essere rifiutata fintantoché ci sono posti liberi per le tariffe minori (quindi, ad esempio, se pago per la tariffa più costosa posso di fatto parcheggiare dovunque ci siano posti liberi, se invece acquisto quella più economica posso parcheggiare solamente in una zona limitata).

Il sistema proposto invece è "intelligente", ossia può comportarsi in maniera differente a seconda dell'esperienza. Come prima cosa, [21] risolve il problema sopra definito supponendo di conoscere esattamente i momenti in cui arrivano le richieste, la loro effettiva durata per ogni tariffa e di conseguenza predire esattamente l'andamento dei processi $E_i(t)$ e $D_i(t)$ $\forall i = 1, \dots, m$, il che rende il problema deterministico. Queste informazioni permettono di risolvere il problema massimizzando, attraverso il calcolatore, la funzione ricavo agendo sull'accettare o meno le richieste che arrivano. Si può ripetere la procedura per diverse condizioni iniziali, da intendersi come andamento delle richieste, e per ogni scenario si hanno a disposizione le scelte migliori da intraprendere. Questo "materiale statistico" verrà poi utilizzato come una specie di *Training set* per elaborare un "criterio" (*Fuzzy Rules*) che permetta in real-time di predire l'andamento delle

richieste ed agire di conseguenza. Il sistema in base all'andamento delle richieste riesce a risalire alla Fuzzy Rule corrispondente, che decide se accettare o meno la richiesta corrente. Nota: ogni classe ha la sua base di Fuzzy Rules. Per non appesantire la trattazione rimando in [22] uno studio che mostra come effettivamente creare queste regole a partire da associazioni input-output.

Per verificare l'efficacia di questa soluzione (*FL solution*) al problema, il modello è stato testato su 10 esempi numerici, che differiscono in base alla struttura del parcheggio o dalla distribuzione degli arrivi/partenze (i tempi interarrivo ed interpartenza hanno distribuzione esponenziale). Prima del test effettivo sono stati creati i diversi set di Fuzzy Rules, usando ulteriori 10 esempi numerici. I risultati sono stati confrontati con quelli ottenuti con altre due strategie: la strategia IP (*Integer Programming*) che dispone dell'andamento delle richieste in anticipo, perciò conosce già che richieste accettare per massimizzare i ricavi (di fatto si tratta di un Upper Bound). La seconda strategia è quella *FIFO*, ossia vengono accettate le richieste man mano che arrivano e si liberano posteggi, senza nessun particolare criterio (se non quello First in-First out). Di seguito una tabella in cui vengono confrontati i ricavi delle tre strategie, mostrando l'efficacia dell'algoritmo proposto.

Problem	Parking capacity	IP solution [S]	FL solution [S]	FIFO solution [S]	Relative error [%] (IP – FL)/IP
1	100	1787.49	1675.17	1514.63	6.28
2	150	2350.26	2217.68	2059.50	5.64
3	250	3647.42	3367.59	3250.38	7.67
4	350	4385.44	4256.81	4148.09	2.93
5	300	3831.10	3724.33	3556.88	2.79
6	120	2340.11	2125.63	1890.95	9.16
7	200	3125.01	3063.64	3044.88	1.96
8	180	3101.62	2962.03	2709.01	4.50
9	350	4860.19	4461.43	4185.12	8.20
10	100	1115.74	1029.09	974.72	7.77

Nota: le prestazioni potrebbero migliorare ulteriormente se si aumenta il bacino di Fuzzy Rules, e per fare ciò è sufficiente aumentare il numero di "esempi" usati per allenare l'algoritmo.

A differenza delle soluzioni viste finora, in [2] e [14] si cambia approccio al problema con l'introduzione della Game Theory. In particolare, in [14] il cambio di approccio viene giustificato dalla maggior sicurezza nella guida di cui i guidatori potrebbero godere. Infatti, l'utilizzo di una app introdotta in [10] potrebbe causare distrazioni del conducente, in quanto sarebbero necessarie continue interazioni con l'applicazione. Sarebbe molto più sicuro che fosse la stessa app che in autonomia scegliesse la destinazione, dove è più probabile trovare parcheggio. Ciò porta però al problema della scelta dell'algoritmo da utilizzare, ed è proprio questo lo scopo del paper ([14]).

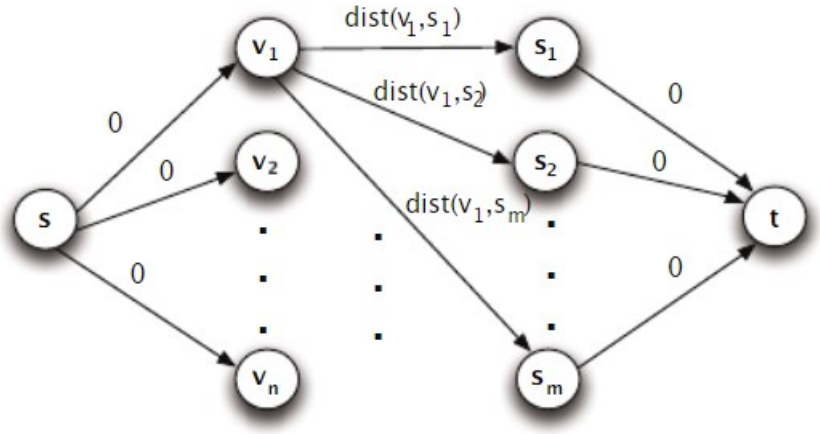
In [14] vengono presentati in realtà due approcci differenti, nel primo è presente una cosiddetta *Authority*, ossia un ente che assegna ai diversi guidatori il posteggio, mentre il secondo consiste in una competizione tra agenti distribuiti non collaborativi.

Vengono poi definiti:

1- due insiemi V ed S, che sono insieme di veicoli e posteggi rispettivamente
2- una funzione distanza $dist: (V \cup S) \times S \to \mathbb{R}$ che rappresenta la distanza tra un veicolo ed un posteggio, o tra un posteggio ed un altro
3- una funzione di *assignment* $g: V \to S$, in cui $g(v)$ rappresenta lo slot assegnato al veicolo v. Nota: se a più di un veicolo viene assegnato lo stesso slot, solamente quello più vicino ad esso potrà parcheggiarvi.

4- il costo associato al parcheggio viene definito come $C_g(v) = dist(v, g(v))$, se v riesce affettivamente a parcheggiare nello slot assegnato, altrimenti varrà $C_g(v)' = C_g(v) + \alpha$, dove α è una penalità.
5- viene definito infine il costo di un *assignment* $C_g = \sum_{v \in V} C_g(v)$

L'algoritmo adottato nel modello *Authority* consiste nel minimizzare C_g per tutte le possibili g, e viene dimostrato avere complessità polinomiale. In realtà, la funzione da minimizzare è leggermente più complicata, infatti le varie distanze tra il veicolo i e lo slot j possono venire pesate per una variabile $y_{ij} \in \{0,1\}$, e ciò può rappresentare una politica di "welfare", ad esempio se i è un veicolo con una persona disabile a bordo e j è uno slot riservato a persone con disabilità, allora y_{ij} potrebbe valere 0. Di seguito una rappresentazione del grafo risultante.



Il modello *Authority*, sebbene rappresenti il compromesso migliore tra tutti gli agenti, comporta che esistano delle situazioni in cui il singolo agente non compia la scelta migliore per sé stesso. Viene infatti analizzato successivamente un modello in cui ogni veicolo, che d'ora in avanti sarà un *player*, conoscerà la posizione di tutti gli atri veicoli ed adotterà la strategia che gli conviene maggiormente, ossia quella che minimizza il **suo** costo che vale $C_g(v)$. Lo scopo è quello di raggiungere l'*Equilibrio di Nash*, ossia un insieme di scelte che portino a ciascun player il *costo* minore possibile in maniera unilaterale: nel senso che, per avere un costo ulteriormente inferiore, sarebbe necessario che i diversi player potessero cooperare tra di loro. L'algoritmo per raggiungere l'equilibrio consiste nell'assegnare allo slot s il veicolo v più vicino in termini di $dist(v, s)$. Viene poi dimostrato che il *Price of Anarchy*, ossia il rapporto tra il costo associato all'equilibrio di Nash e il costo migliore possibile non è limitato, perciò l'effettuare scelte egoiste ed il disporre della posizione di tutti i players da parte di ogni agente può portare ad un costo totale arbitrariamente grande.

Il secondo modello che viene analizzato consiste in un Game in cui non vi sono scambi di informazioni tra i player, perciò oltre a non essere a conoscenza della posizione degli altri ciascun agente v non conosce nemmeno il proprio costo, associato al parcheggiare nel generico slot s. Le uniche informazioni che i players avranno saranno gli slot liberi: questo perché le scelte che verranno introdotte si basano su considerazioni probabilistiche, e ciascun veicolo effettuerà le scelte che minimizzano il valore atteso del costo. A differenza delle precedenti analisi, ogni game termina quando un singolo veicolo trova parcheggio, poi verrà lanciato un nuovo game con *n-1* veicoli ed *m-1* posteggi, ciò porta ad una nuova definizione di costo e penalità. Il costo per il veicolo v sarà la

il valore atteso della distanza percorsa per raggiungere il suo posteggio target s, ed in caso non sia libero verrà aumentato di α. In questo caso α è il valore atteso della distanza dal veicolo allo spot più vicino tra gli *m-1* rimasti (così da premiare i veicoli che si sono posizionati meglio per successive iterazioni).

Nel primo scenario analizzato ci sono *n*=2 player ed *m*=3 slots in linea, ossia $s_1 = 0$ sta a sinistra mentre $s_2 = s_3 = 1$ sono slots che stanno a destra. Ogni player parte da una posizione x distribuita uniformemente nell'intervallo [0,1], e sceglierà di cercare a destra con probabilità $p(x)$ oppure a sinistra con probabilità $1 - p(x)$. In questo contesto, la scelta ottima che conduce all'equilibrio di Nash consiste in un'opportuna scelta di $p(x)$. Viene poi dimostrato che la scelta ottima $p^*(x)$, con queste condizioni, vale:

$$p^*(x) = \begin{cases} 1 & \text{if } x \leq 3/8 \\ 0 & \text{if } x > 3/8 \end{cases}$$

(1) *Strategia ottima*

in quanto minimizza il valore attesto del costo $C^*(x)$. Nota: qui $C^*(x)$ è funzione solamente della posizione, visto che entrambi i giocatori adottano la medesima strategia.

Successivamente vengono analizzate le prestazioni di un'altra strategia, che a prima vista può sembrare la più ragionevole, ossia:

$$p(x) = \begin{cases} 1 & \text{if } x \leq 1/2 \\ 0 & \text{if } x > 1/2 \end{cases}$$

(2) *Strategia intuitiva*

L'adottare questa $p(x)$ porta ad un peggioramento del 4.5% in termini del costo atteso, ma si rivela efficace nel caso *m*=2.

Il problema delle precedenti strategie è la complessità di calcolo del valore atteso in caso di n ed m arbitrari. L'ultima strategia che viene analizzata è di tipo euristico, e prende spunto dalle due precedenti $p(x)$ e $p^*(x)$. Lo spostarsi a destra o a sinistra dipende dalla posizione iniziale del veicolo, ma la posizione che fa da "spartiacque" in cui scelgo 0(sx) o 1(dx) dipende dalla distribuzione degli slots: se ho due slots vale 1/2, mentre se ho due slot a destra ed uno a sinistra vale 3/8. Se ci immaginiamo gli slots come dei pianeti, allora i valori limite potrebbero essere visti come i centri di massa del sistema, e ciò che viene attirato è la probabilità di convergere a destra o a sinistra. Questa intuizione porta al Gravitational Model of Parking, in cui ogni veicolo si muoverà verso la maggior "concentrazione di parcheggi liberi nello spazio". La "forza" che s esercita su v viene espressa dalla formula:

$$F(s, v) = 1/dist(s, v)^{\beta}$$

dove β è una costante da determinare sperimentalmente via simulazione. Viene definita inoltre una velocità z, e ad ogni step di simulazione il veicolo si muoverà di z unità in direzione $\bar{F}$. Una volta che $dist(s, v) < z$ per un certo $s \in S$, v parcheggerà in s, che per ipotesi è sicuramente libero visto che ad ogni parcheggio l'insieme S degli spots liberi viene riaggiornato. L'adottare questa strategia ha portato in tutti casi ad una significativa diminuzione del costo rispetto alla strategia intuitiva (2), (che generalizzata ai casi multidimensionali consiste nel tentare nello spot più vicino).

Il lavoro svolto in [2], a differenza di [14], studia il problema del trovare parcheggio in uno scenario di Parcheggio Multilivello. Benché in fin dei conti sia possibile adattare il modello di [14] anche all'analisi di [2], ad esempio sostituendo i nodi spot con vettori che rappresentano i posteggi dei diversi piani e pesando i diversi link a seconda del piano a cui fanno riferimento, l'approccio adottato è differente. L'analisi di [2] infatti è prettamente empirica, non è presente infatti un'analisi matematica rigorosa, questo per la complessità dello scenario: anche in [14] in realtà, una volta rilassati i vincoli di simulazione, quindi molti agenti e molti posteggi, si è abbandonata la via più rigorosa a fronte di una sperimentale. Il modello adottato in [2], come sarà chiaro nella sezione successiva, è simile a quello che verrà adottato nel presente lavoro: ossia il parcheggio verrà modellato con L livelli, e sarà possibile spostarsi solamente nel livello adiacente all'attuale comportando un costo pari 3. Sarà possibile, inoltre, tentare la ricerca nel piano corrente, comportando un costo pari a 7. Nella prima parte del paper le strategie che vengono analizzate sono *a priori*, perciò ciascun veicolo che entra nella struttura segue un percorso prestabilito di ricerca: questa scelta può venire giustificata dal fatto che i diversi agenti potrebbero essere anche veicoli a guida autonoma. Per evitare simulazioni di durata indefinita, cosa che può capitare per alcune strategie, è stato posto un limite di tempo alla simulazione: i player ancora attivi, dopo che è scaduto il tempo, riceveranno una penalità.

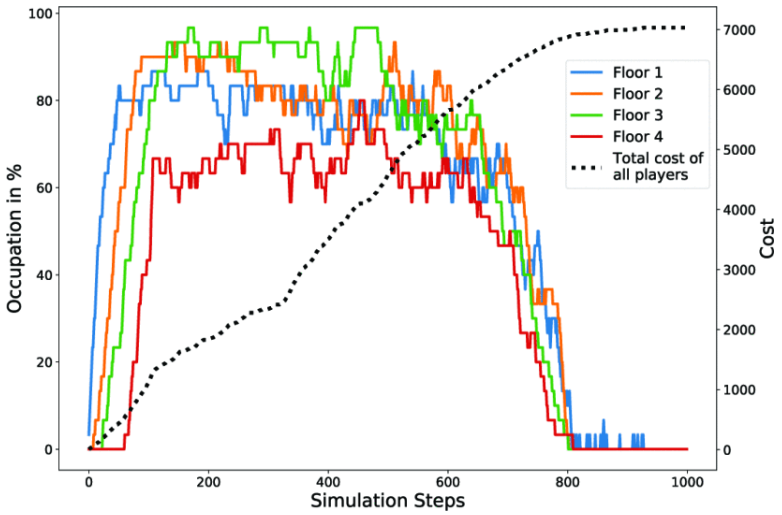
Gli agenti, nella prima analisi, non conoscono né l'orario né l'ordine di ingresso, il che rende il Game completamente simmetrico. L'ordine di entrata (per ipotesi, non sono ammesse entrate multiple nello stesso istante) di conseguenza non è rilevante ai fini della scelta della strategia da adottare, ma lo è dal punto di vista dei costi. Infatti, il primo agente che entra troverà il parcheggio vuoto e sarà avvantaggiato rispetto ai successivi: perciò è stato introdotto un player speciale, chiamato Nature, responsabile dell'ordine di ingresso (quindi se N è il numero totale di players, ognuno di essi avrà probabilità 1/N di occupare una certa posizione nella coda di ingresso). Con queste ipotesi, e diverse simulazioni col variare di N ed L, segue la tabella con i risultati delle strategie *a priori* in ordine decrescente di prestazioni. Nota: S=Search, U=go Up, D=go Down.

strategy	utility value
SUSUSUS	-5.2
SUSUUSDS	-5.6
SUUSDSUUS	-6.0
SUUSUSDDS	-6.4
USDSUUSUS	-6.4
USDSUUUSDS	-6.8
USUSDDSUUUS	-6.8
SUUUSDDSUS	-7.2
SUUUSDSDS	-7.2
USUSUSDDDS	-7.2

Come si può notare, la strategia migliore sembra essere una semplice scansione sequenziale del parcheggio.

La seconda parte del paper arricchisce il modello, simulando così una classica giornata in cui la variabile tempo gioca un ruolo chiave. La giornata viene modellata come una successione di 1000

istanti temporali, ed è l'agente Nature a decidere ad ogni istante se avviene un arrivo o una partenza. Per rendere più realistica la simulazione, la probabilità di avere arrivi nell'istante t non rimane costante per tutta la simulazione, bensì segue la seguente legge: $P_{arrivo} = 1 - t/1000$, perciò passa dal 100% (inizio della giornata) fino a raggiungere lo 0%. Infine, si assume che sia il numero di veicoli che possono entrare nella struttura, sia la popolazione totale di veicoli della simulazione siano limitati. Di seguito un'immagine che mostra come varia il livello di riempimento del parcheggio nel corso della giornata.

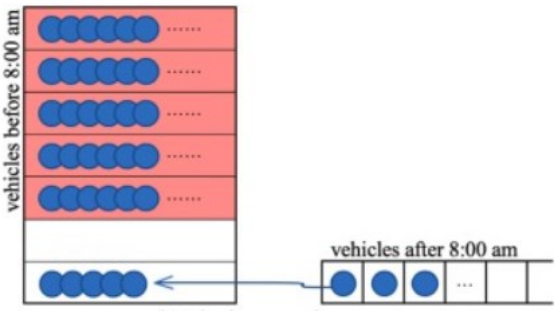


Dopo l'introduzione del nuovo modello sono state analizzate due ulteriori strategie, la prima, a cui per semplicità mi riferirò col nome di Mobility Test Strategy, consiste in una differenziazione degli agenti in base alla loro propensione alla ricerca. Ogni agente possiederà un campo *mobility value* cha va da 0 ad 1: maggiore sarà il valore più sarà probabile che l'agente decida di cambiare piano e non cercare in quello corrente. La seconda (Time Strategy) invece sfrutta l'informazione aggiuntiva introdotta dal modello, ossia l'orario di ingresso. I vari agenti quindi dispongono non solo del tempo in cui entrano nella struttura, ma sanno anche come si comportano gli altri agenti. Infatti, all'inizio della giornata si suppone che il parcheggio sarà libero, quindi si prediligerà la ricerca nei primi livelli mentre man mano che il tempo passa si prediligeranno livelli successivi.

In conclusione, viene dimostrato che la Mobility Test Strategy funziona meglio con una equa distribuzione di *mobility value* tra gli agenti. Sfruttare invece l'informazione temporale aggiuntiva si riflette in un miglioramento dei costi. Bisogna puntualizzare però che il successo della Time Strategy si basa sulla conoscenza del comportamento degli altri players, in particolare che tutti adottino al stessa politica, cosa che in pratica potrebbe non essere verificato. La strategia migliore tra tutte rimane comunque la scansione sequenziale del parcheggio.

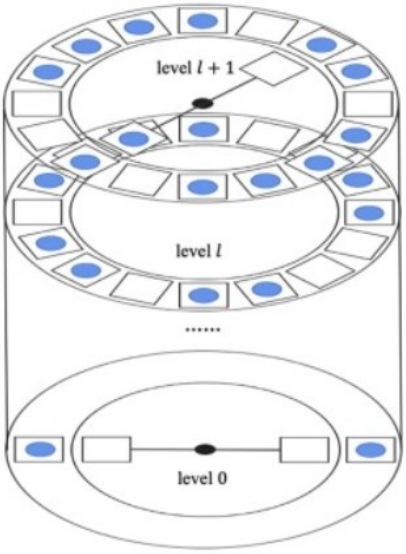
Un'ultima tipologia di struttura viene analizzata in [23], e consiste in un parcheggio multilivello completamente automatizzato. L'introduzione di parcheggi di questo tipo porta a due benefici essenzialmente. Il primo beneficio è uno sfruttamento dello spazio disponibile più efficiente, non essendo necessari nessun tipo di rampe d'accesso o di spazi aggiunti per aprire le portiere delle auto per scendere dal veicolo (viene riportato che l'area media per ogni posteggio è di $21m^2$, invece quella dei parcheggi multilivello "normali" con rampe varia dai $30m^2$ ai $42m^2$). Il secondo

beneficio invece è un minor spreco di tempo da parte degli automobilisti nel cercare parcheggio, dato che è sufficiente lasciare la macchina all'ingresso (ci penseranno poi gli appositi ascensori a parcheggiare l'auto). I problemi di questo sistema, che [23] punta a risolvere, sono anch'essi di due tipi: l'utilizzo efficiente degli ascensori e l'assegnamento dei posteggi. Sul secondo punto viene fatto un ragionamento simile a [2], ossia un comportamento diverso a seconda della fase della giornata. Una prima soluzione proposta potrebbe essere quella di riservare i posteggi più vicini all'ingresso all'orario di punta, cosicché quando c'è l'afflusso massimo di utenti il tempo di attesa in coda venga ridotto il più possibile.



Schema strategia efficiente (ora di punta 8-9) *Parcheggio automatico ed ascensore*

Il parcheggio verrà strutturato come segue: avrà L livelli, e gli utenti lasceranno il proprio veicolo al livello 0. I livelli saranno di forma circolare, e i posteggi saranno uniformemente distribuiti. Sarà presente un solo ascensore che possiederà M piastre grazie alle quali potrà prelevare e/o depositare i veicoli simultaneamente. Il movimento dell'ascensore è periodico, ossia carica i veicoli al livello 0 poi sale fino all'ultimo livello ed infine torna giù, nel mentre avrà parcheggiato **tutti** i veicoli. Il ritiro segue una politica di scheduling, aggiornata in tempo reale con l'obiettivo di essere più efficiente possibile.

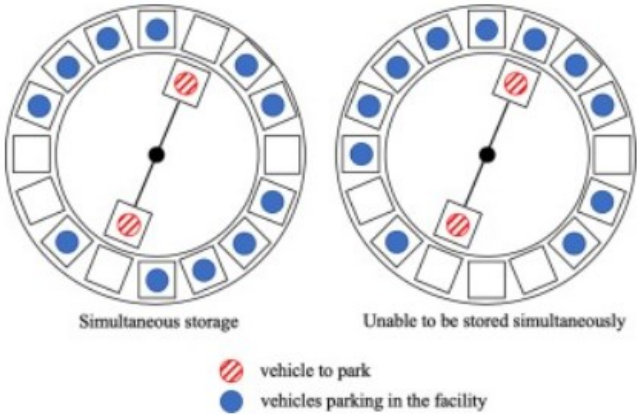


Schema interno del parcheggio (M=2)

Per ottimizzare l'assegnamento dei livelli, tenendo in considerazione la natura stocastica del problema, si è adottato per il cosiddetto "Reinforcement Learning", e cioè una tecnica che permette al sistema di "imparare" interagendo con l'ambiente in cui opera. Senza scendere troppo nel dettaglio (per trattazione più approfondita, si rimanda a [23] sez.3.2), conoscendo esattamente lo stato dello scenario e dati gli insiemi di scelte possibili, il sistema seguirà una certa politica iniziale.

Successivamente, mediando i risultati ottenuti applicando la stessa scelta allo stesso scenario, sarà possibile fornire una valutazione della politica, quindi delle scelte che si stanno intraprendendo (*policy evaluation*). Il sistema può inoltre migliorare la politica corrente, effettuando una scelta *greedy*, quella scelta cioè che si aspetta porti ad una utility migliore (*policy improvement*); verrà poi valutato se ciò ha portato effettivamente ad un miglioramento. Il fatto di adottare scelte differenti nel tentativo di migliorare la strategia corrente, pone il problema di quando sfruttare le informazioni ottenute, quindi tenere la politica corrente, o quando cambiarla. Ciò viene risolto introducendo una variabile aleatoria, che con una certa probabilità (variabile) possa fare "esplorare" all'algoritmo soluzioni differenti (ad esempio, nella fase di training la probabilità è maggiore). Il tutto si arresta quando, per ogni scenario, non vi sono scelte alternative migliori da quelle previste dalla politica corrente.

Dato quindi il livello in cui parcheggiare l'auto, bisogna scegliere il posteggio adatto. I posteggi vengono scelti in modo tale da usare in maniera più efficiente possibile l'ascensore, il che si traduce nel poter parcheggiare simultaneamente più veicoli possibili.



Lo studio si conclude con tre simulazioni dove vengono variate sia la distribuzione degli arrivi, (vengono introdotti anche degli orari di punta) che la durata media di sosta dei veicoli all'interno della struttura. Per i diversi scenari proposti, l'algoritmo, basandosi sulle informazioni acquisite, riesce a stabilire strategie appropriate ottenendo in due scenari su tre un miglioramento delle prestazioni significativo rispetto ad una strategia di confronto, che prevede il parcheggio nel primo livello libero. Vengono infine riportati alcuni suggerimenti per migliorare ulteriormente l'efficacia del sistema, come ad esempio il disporre di informazioni maggiori riguardo il comportamento degli utenti (un'ulteriore conferma dell'importanza dei dati sottolineata in [11]) oppure uno studio più approfondito sul layout della struttura.

Sezione II: Definizione del Modello

Assumiamo che il parcheggio sia formato da L livelli tutti uguali, ciascuno dei quali è costituito da $nPark$ posteggi. Ciascun veicolo entra al livello $\ell = 1$, con $1 \leq \ell \leq L$ e può scegliere di cercare nel piano attuale, o spostarsi di piano. Il costo associato allo spostarsi in un piano adiacente a quello attuale è $Cm = 3$, mentre il costo associato alla ricerca in un qualsivoglia piano è $Cs = 7$. La scelta dei valori di Cm e Cs è arbitraria e non univoca, tuttavia la scelta $Cs > Cm$ è giustificata dal fatto che la ricerca di un parcheggio in un piano ha un costo in termini di tempo maggiore rispetto al tempo speso per spostarsi nel piano adiacente, specialmente se il piano ha un numero elevato di posteggi. Nessun agente, se non diversamente indicato, ha alcuna informazione riguardo lo stato del parcheggio.

Modello di costo ed Utility Function

Il modello di costo che ho deciso di adottare prende spunto da quello adottato in [2] e tiene conto sia del tempo impiegato per la ricerca che del livello in cui l'autoveicolo trova parcheggio. In particolare, definisco la seguente funzione di costo:

$$costo = (n + i)Cm + kCs$$

dove:

$n = numero\ di\ piani\ cambiati$: in fase di ricerca, l'agente deve scegliere che piano tentare. Spostarsi di piano come accennato in precedenza ha costo $Cm = 3$ in caso mi sposti nel piano adiacente, se decido di spostarmi in un altro piano il costo sarà multiplo di Cm

$k = numero\ di\ tentativi$: fare una ricerca in un piano ha associato un costo di Searching pari a $Cs = 7$, e non effettuo una ricerca in ogni piano cambiato. Se passo dal primo al quinto piano cambio quattro piani, ma cerco solamente al quinto, facendo perciò un solo tentativo.

$i = indice\ dell'ultimo\ piano\ visitato$: a differenza del modello adottato in [2], ho deciso di introdurre quest'ulteriore parametro che rappresenta il piano in cui il veicolo ha trovato parcheggio. Gran parte degli automobilisti vorrebbe parcheggiare vicino all'ingresso, per poi essere vicino all'uscita, che assumo si trovi nello stesso piano dell'ingresso, ed evitare di perdere tempo nel salire (o scendere) di livello. Tengo già conto di ciò in realtà con nCm, visto che salire di piano comporta un costo maggiore. Quello di cui in [2] non si tiene conto è del tempo di uscita dal parcheggio, il seguente esempio dovrebbe chiarire il tutto. Supponiamo ci sia un veicolo al quarto piano, e viene a sapere in qualche modo che ci sono due posti liberi: uno al quinto ed uno al primo piano. Se dovesse scegliere in base al costo minore, e non tenesse conto di i, sceglierebbe il quinto piano: spende solamente un Cm per salire di piano, invece per tornare al primo spenderebbe $3Cm$. In realtà il veicolo dovrà prima o dopo uscire, e quindi tornare al primo piano, perciò il costo totale di scegliere il quinto piano sarebbe $Cm + 5Cm$ a fronte dei $4Cm$ (per semplicità ho fissato il costo di uscita dal primo piano pari a Cm). Perciò conviene comunque scendere al primo!

Nel caso un veicolo non trovi parcheggio, il costo associato alla ricerca sarà solamente quello temporale, in altre parole $i = 0$. Terrò comunque traccia del numero di veicoli che non trovano parcheggio, e sarà un parametro di notevole importanza per confrontare le diverse politiche di decisione.

In Game Theory, un'Utility Function è una mappa che, dato l'insieme di decisioni che l'agente compie, fornisce un valore che in qualche modo quantifica la bontà delle scelte intraprese. Va da sé che per una corretta modellazione debbano venire presi in considerazione tutti gli aspetti del problema. La scelta dell'Utility Function si basa sul costo, infatti l'idea è che sia decrescente con l'aumentare del costo: maggiore è il costo minore sarà l'Utility.

Definisco quindi la seguente Utility Function:

$$Utility\ Function = -\frac{costo}{10}$$

con questa definizione, l'Utility Value massima vale -1 (parcheggio al primo livello al primo tentativo).

Modello temporale, processo degli arrivi e delle partenze

La giornata viene modellata come una sequenza crescente di 1000 istanti temporali, con un istante $t_{punta} = 500$ che rappresenta l'orario di punta: la probabilità di avere arrivi nell'istante $t \in [1{,}1000]$ è crescente man mano che t si avvicina a t_{punta}. Il numero di veicoli che entrerà nel parcheggio è fissato a priori.

I processi di arrivi e partenze sono di Poisson, con tempi di interarrivo esponenziali di media λ e μ rispettivamente, questi parametri subiranno una variazione man mano che mi avvicino all'orario di punta: all'inizio della giornata per $t \in [1{,}250]$ varranno infatti $\lambda = 0.8$ e $\mu = 1$, mentre poi varrà $\lambda = 0.625$. In entrambi i casi il sistema è instabile, ma la coorte limitata di veicoli fa sì che il riempimento completo del parcheggio la raggiunga raramente e solamente nell'intorno dell'orario di punta, e per breve periodo. Ho optato per questa decisione perché, per testare la bontà delle politiche, necessito di un livello alto di arrivi ed occupazione. Avere un sistema stabile con $\lambda < \mu$, anche se tendente all'instabilità, mi richiederebbe un numero di veicoli e di tempo molto elevati. Ciò si tradurrebbe in simulazioni molto pesanti in termini di tempo di calcolo.

Gli arrivi e le partenze sono singoli, non possono avvenire quindi arrivi e/o partenze in contemporanea. Di fatto ho modellato i processi di arrivi e partenze come una Catena Markoviana, come fatto in [11]. Questa è sicuramente un'astrazione della realtà, visto che arrivi e/o partenze possono benissimo avvenire nello stesso momento; tuttavia, l'obiettivo è valutare come diverse politiche di scelta possano inficiare i costi di ricerca, perciò quest'astrazione non pregiudica la consistenza dei risultati ottenuti. Inoltre, nello stesso istante solo un veicolo è ammesso nel parcheggio: anche questa è un'astrazione della realtà, e perciò non tengo conto delle situazioni in cui ci sono più contendenti nello stesso istante ad effettuare una ricerca nello stesso piano. Anche qui però con il modello di costo adottato è completamente indifferente a questo genere di situazioni.

Le partenze inoltre sono distribuite uniformemente in tutti i piani ed **indipendenti dall'arrivo dei veicoli.** Questa decisione può benissimo essere cambiata in futuro, magari adattando la simulazione ad un parcheggio reale in cui si hanno a disposizione statistiche sul comportamento degli utenti sufficienti.

Infine, un agente che ha già esplorato tutti i piani rimane comunque all'interno della struttura finché non esaurisce il numero dei tentativi, a meno che non disponga di ulteriori informazioni (sez.V).

Sezione III: Strategie a priori

Le prime strategie che ho deciso di studiare sono le più semplici ed anche molto diffuse: ogni autoveicolo adotta la stessa strategia e decide quindi che piani ha intenzione di visitare a priori, quindi indipendentemente dall'esito dei tentativi, dal comportamento degli altri autoveicoli o dal tempo di arrivo.

Tratterò in dettaglio la strategia *SUSU* (Search,Up,Search,Up) che consiste semplicemente nel tentare il piano corrente, e se occupato salire al piano successivo. Una volta raggiunto l'ultimo piano si riparte dal primo. Ho deciso di dare importanza a questa strategia rispetto alle altre dello stesso tipo per due motivi:

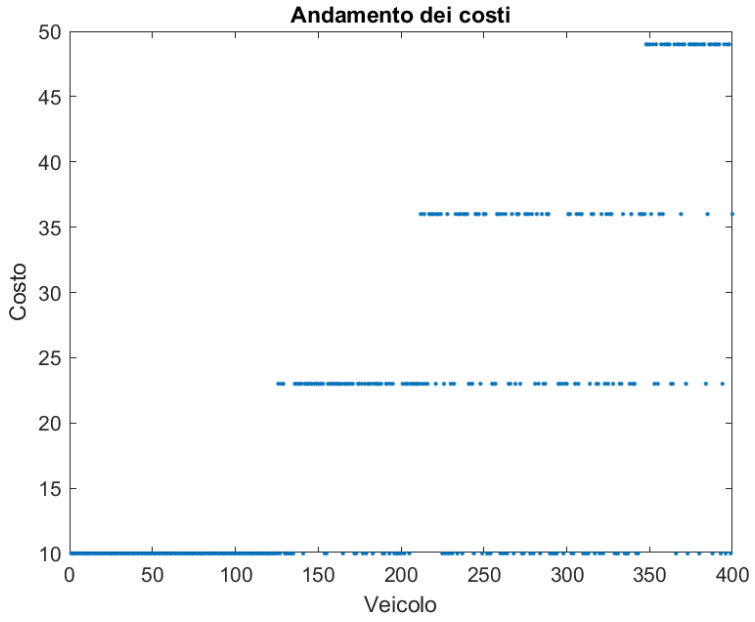
1) Secondo il lavoro svolto in [2], dovrebbe essere la migliore tra tutte le strategie a priori. Vediamo se con il mio modello questo risultato viene confermato o meno.
2) Permette di evidenziare alcune caratteristiche interessanti dell'effetto del numero di piani, di utenti e le probabilità di arrivo e di partenza sul costo medio di ricerca. Caratteristiche che poi si ripercuoteranno in tutte le strategie, quindi preferisco affrontarle ora.

Politica SUSU

Parametri di simulazione:

$$L = 4 \quad nPark = 20 \quad nAgents = 400 \quad \max\_attempts = 10$$

Come prima cosa, faccio presente che il numero di agenti è molto superiore al numero di posti disponibili, ma grazie alle partenze, si possono liberare dei posti precedentemente occupati. Inoltre, l'andamento dei costi è crescente man mano che passa il tempo: dopo una prima fase in cui i veicoli accumulano il costo minimo possibile, perché trovano posto al primo tentativo al primo livello, esso aumenta. Il massimo lo si raggiunge per gli ultimi veicoli, che sono costretti a cercare parcheggio al quarto livello non prima di aver cercato in tutti i livelli precedenti.



Costi per veicolo-politica SUSU

Possiamo notare che il minimo dei costi è pari a 10 (primo tentativo-primo piano). Inoltre, il fatto che il costo massimo sia di poco inferiore a 50, precisamente 49, ci suggerisce che ciascun veicolo ha trovato parcheggio visitando ogni piano solamente una volta. Di conseguenza il quarto piano è rimasto al di sotto della capienza massima. Questa intuizione è confermata dal seguente grafico, che illustra lo stato di riempimento dei livelli in funzione del tempo.

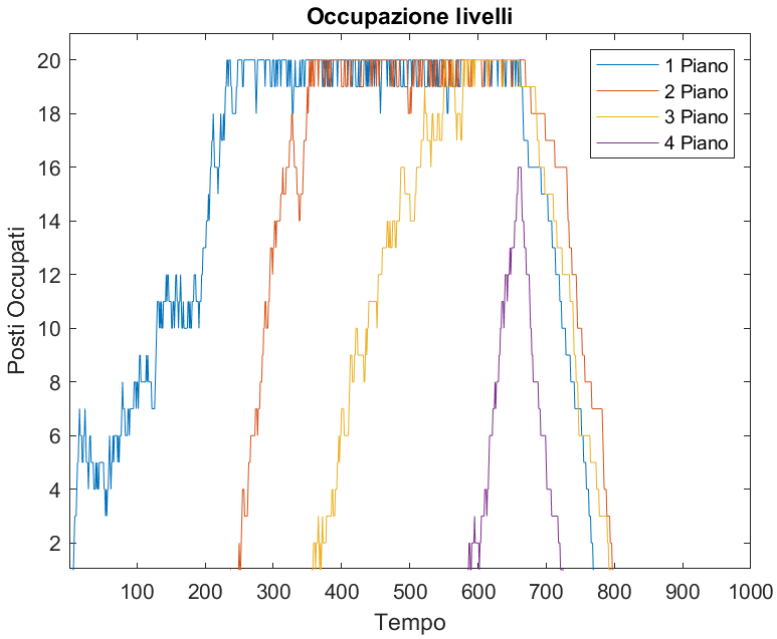


Fig. Simulazione 1

Possiamo notare un'altra caratteristica di questa strategia, ossia che i livelli vengono riempiti in successione: riflette giustamente l'andamento dell'ordine di visita definito a priori. Inoltre, possiamo notare che abbiamo una densità di arrivi minore fino a $t = 250$, poi aumenta per poi crollare quando abbiamo terminato il numero di veicoli.

I due grafici successivi mettono in evidenza il legame tra λ e μ con il livello di congestione del parcheggio.

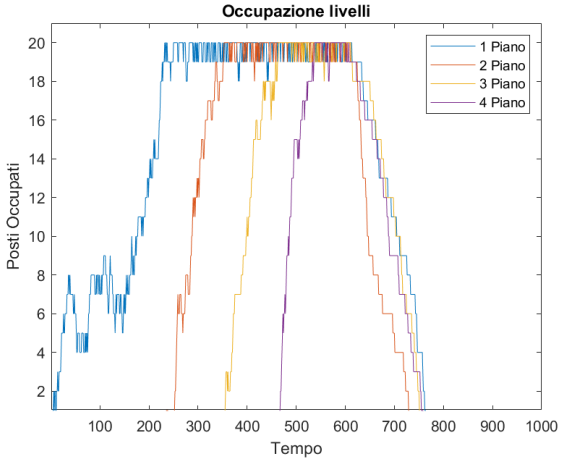


Fig. Simulazione 2

$\lambda = 0.45, \mu = 1$

Utility Media= -2.10

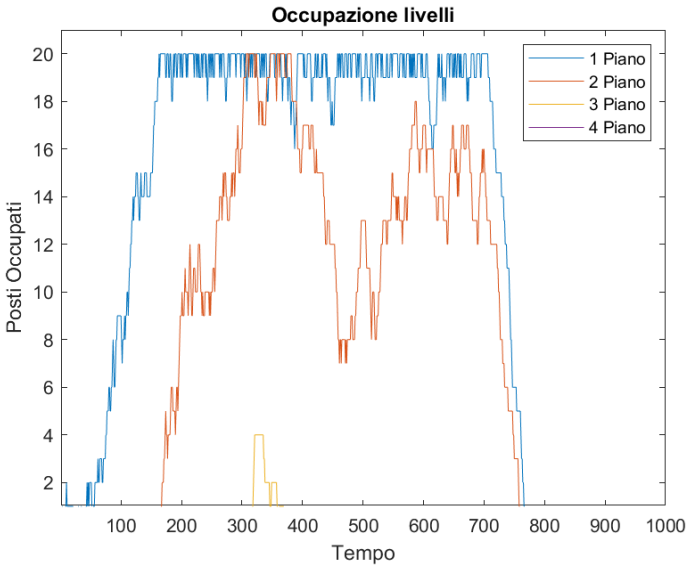
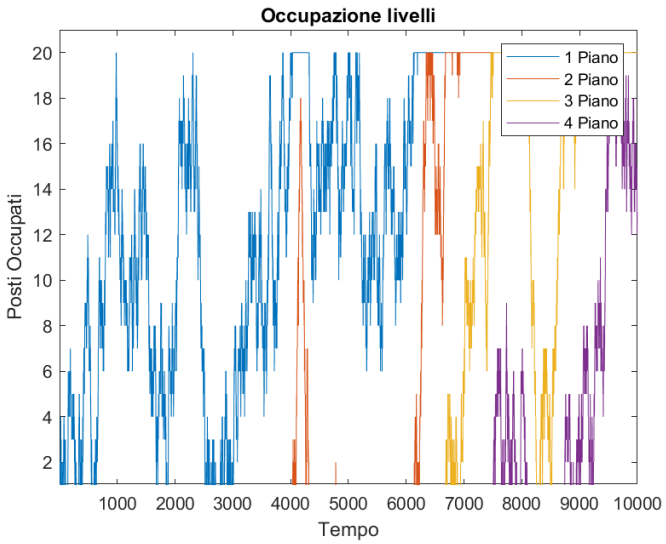


Fig. Simulazione 3

$\lambda = 0.8,\ \mu = 1$

Utility Media = -1.53

Riporto ora un ulteriore grafico, che dimostra che ponendo $\lambda = \mu = 1$ abbiamo comunque instabilità. Per la simulazione ho usato 10000 veicoli e 10000 istanti temporali, e ciò ha richiesto qualche minuto di calcolo.



Variazione dei parametri strutturali

Analizzo ora cosa capita variando il numero di piani. Per fare ciò mantengo costante la capienza totale del parcheggio, mantenendo costanti gli altri parametri.

1) Aumento del numero di piani (di conseguenza, minor numero di posti per ogni piano)

$L = 8$ $nPark = 10$

Utility Media = -2.82 $Abbandoni = 0$

2) Diminuzione del numero di piani

Per completezza riporto anche i risultati per un numero di piani inferiore, i cui risultati sono speculari al caso precedente.

$L = 2$ $nPark = 40$

Utility Media = -1.38 $Abbandoni = 0$

Aumentando invece il numero di agenti, aumentiamo il traffico totale: ciò comporta che nell'intorno di t_{punta} abbia sempre veicoli a disposizione, che congiuntamente ad una probabilità di arrivo elevata, causano il sovraffollamento del parcheggio.

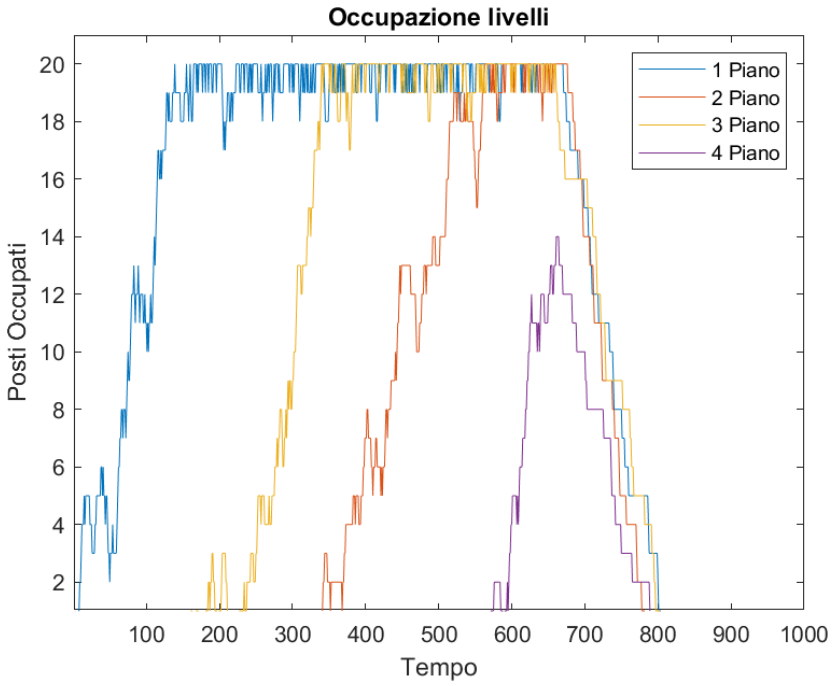
Ulteriori strategie a priori e confronto

Dopo aver analizzato in dettaglio cosa capita ai costi variando i vari parametri di simulazione, d'ora in avanti utilizzerò i seguenti per confrontare le diverse strategie a priori:

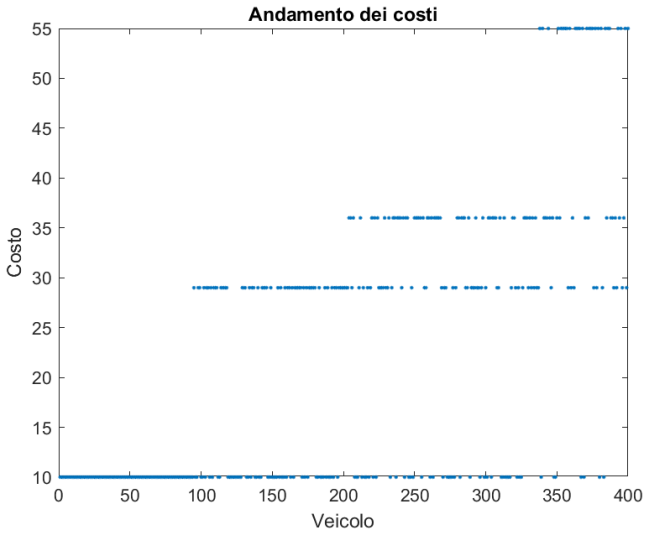
Parametri di simulazione:

$$L = 4 \quad nPark = 20 \quad nAgents = 400 \quad \max\_attempts = 10 \quad \lambda = 0.625 \quad \mu = 1$$

1- Strategia SUUSD (Search,Up,Up,Search,Down): questa prima strategia consiste nel cercare al primo piano; in caso di esito negativo salire di due piani e cercare nel terzo per poi eventualmente scendere e cercare nel secondo, e così via. In particolare, l'ordine di visita sarà: [1 3 2 4 3 1 3 2 4 3]. L'idea che sta alla base di questa strategia è che, in caso di fallimento sul piano i-esimo, mi convenga salire di due piani ipotizzando che anche il successivo sia occupato.



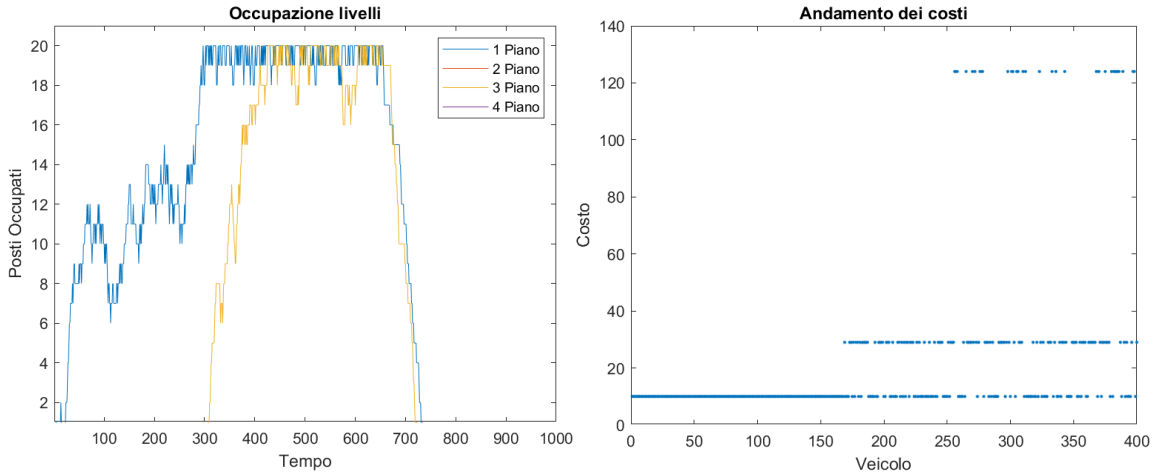
Possiamo notare che non questa strategia i piani maggiormente occupati saranno il primo ed il terzo, come da aspettative visto che per ora tutti gli agenti usano la stessa strategia.



$$Utility\ Media = -2.30$$

$$Abbandoni = 0$$

2- Strategia SUUS (Search,Up,Up,Search): questa strategia invece consiste nel cercare al primo piano; in caso di esito negativo salire di due piani e così via. In particolare, l'ordine di visita sarà: [1 3 1 3 1 3 1 3 1 3]. Questa strategia, a prima vista è molto inefficiente visto che non esploro tutti i piani, ma è comunque interessante studiarla in un contesto di agenti che usano tutti la stessa strategia ad uno in cui si possono usare politiche diverse. Infine, essendo una strategia molto particolare, permette di verificare almeno in parte la correttezza del codice.



Come ci aspettavamo, gli unici piani che sono stati occupati sono il primo ed il terzo.

$$Utility\ Media = -2.4$$

$$Abbandoni = 8$$

Essendoci abbandoni, il costo massimo vale $costo = 18Cm + 10Cs$=124 come suggerisce il grafico.

Potrei definire molte altre strategie analoghe alle precedenti, tuttavia preferisco valutare alcune strategie già studiate in [2] per confrontare i risultati lì ottenuti anche col mio modello. Per valutare le varie strategie, quello che faccio è mediare l'Utility Media (di una singola simulazione) ottenuta facendo 40 simulazioni differenti.

Le strategie che andrò a valutare sono:

1- Strategia SUSU che corrisponde all'ordine di visita: [1 2 3 4 1 2 3 4 1 2]
2- Strategia SUUSD che corrisponde all'ordine di visita: [1 3 2 4 3 1 3 2 4 3]
3- Strategia SUUS che corrisponde all'ordine di visita: [1 3 1 3 1 3 1 3 1 3]
4- Strategia SUSUUSDS che corrisponde all'ordine di visita: [1 2 4 3 1 2 4 3 1 2]
5- Strategia USUSUSDDDS che corrisponde all'ordine di visita: [2 3 4 1 2 3 4 1 2 3]

Tabella 1: confronto di Utility Values di diversi agenti con la stessa strategia

Strategia	Utility	N. medio Abbandoni
SUSU	-1.90	2
SUUSD	-2.05	1.28
SUUS	-2.44	31
SUSUUSDS	-1.98	1.43
USUSUSDDDS	-2.46	0.53

Il problema delle strategie soprastanti, è che sono completamente deterministiche, quindi se tutti i veicoli adottano la stessa ne risulta un costo maggiore. Tuttavia, è possibile simulare cosa succede se veicoli adottano strategie a priori differenti in un'unica simulazione. C'è da fare però una distinzione: se per ipotesi i primi N veicoli seguissero la stessa strategia, e poi i successivi $nAgents - N$ veicoli seguissero una strategia diversa, i risultati che otterrei sarebbero diversi rispetto ad un'entrata completamente casuale. A conferma di ciò, ho deciso di simulare entrambe le possibilità.

Tabella 2: Risultati per Agenti che seguono strategie differenti, ordine di entrata rilevante

N. veicoli SUSU	N. veicoli SUUSD	N. veicoli SUUS	Utility	N. medio Abbandoni
200	200	0	-2.0	0.12
200	0	200	-2.41	30.8
300	0	100	-2.57	33
300	100	0	-1.91	0.8

I risultati che otteniamo in questa maniera sono una specie di "media pesata" ottenuta combinando i risultati della Tabella 1 e pesandoli per il numero di veicoli che seguono la stessa strategia.

Per le seguenti simulazioni invece genererò veicoli che seguono diverse politiche con una certa probabilità, cosicché l'ordine di ingresso non giochi un fattore determinante nel costo. Per la legge dei grandi numeri, la probabilità coincide con la frazione dei veicoli (in realtà ciò è vero asintoticamente, quello che faccio è quindi un'approssimazione che però è giustificata dal grande numero di veicoli) che seguono la stessa strategia. Ho deciso di tenere le stesse proporzioni già riportate in Tabella 2, così da evidenziare le differenze tra i due approcci.

Tabella 3: Risultati per Agenti che seguono strategie differenti, ordine di entrata NON rilevante

Pr(veicolo SUSU)	Pr(veicolo SUUSD)	Pr(veicolo SUUS)	Utility	N. medio Abbandoni
0.5	0.5	0	-2.0	2.7
0.5	0	0.5	-2.1	14.7
0.75	0	0.25	-1.9	1.1
0.75	0.25	0	-2	7.7

Sorprendentemente, dopo varie simulazioni, la politica migliore è la combinazione della politica SUSU e USUSUSDDDS al 50%-50% con ingresso non rilevante. Nonostante secondo la Tab. 1 la strategia USUSUSDDDS sembri la più inefficiente, in realtà è l'unica che non esplora il primo piano come primo tentativo. In conclusione, le prestazioni migliori ottenibili combinando strategie a priori sono:

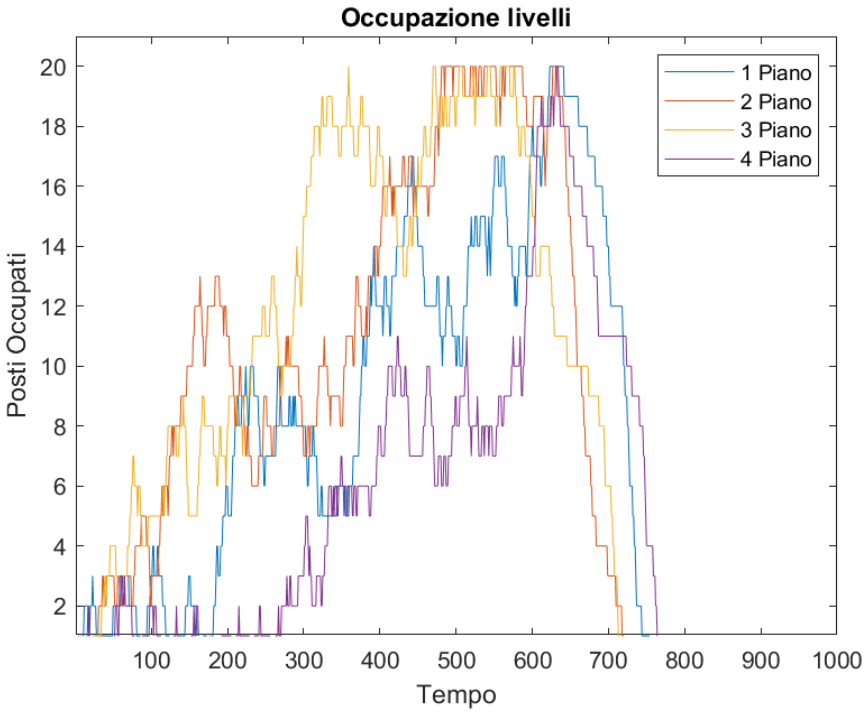
Utility Media = -1.84 N. medio Abbandoni=1.1

Sezione IV: Strategie Probabilistiche

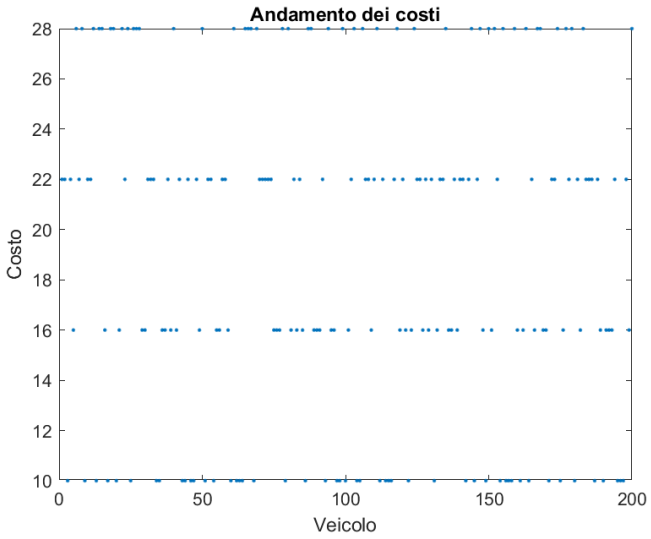
L'aver introdotto agenti che adottano differenti politiche con una certa probabilità ha provocato un costo medio ed un numero di abbandoni minori. Questo risultato ci suggerisce che, al fine di massimizzare l'Utility, il fatto che agenti diversi attuino scelte diverse è cruciale. Perché non estendere quindi l'approccio probabilistico non solo nella scelta della politica degli agenti, ma anche nella politica stessa? Quest'intuizione porta allo studio delle cosiddette Strategie Probabilistiche, in cui i livelli da visitare non sono decisi a priori, ma sono decisi da ciascun agente in maniera casuale nel corso della simulazione. In questa tipologia di strategie non è possibile ricercare nuovamente nel piano appena visitato, ma in quelli visitati in precedenza si.

Strategia Random

La politica più semplice che ho deciso di testare è quella di cercare con probabilità uniforme tra i piani. I parametri utilizzati nella simulazione sono gli stessi delle strategie a priori, così da poter confrontare i risultati ottenuti. Nonostante questa strategia possa sembrare completamente inefficiente, visto che salgo o scendo indifferentemente di un numero casuale di piani, è quella che fornisce ai vari agenti la più alta incorrelazione di scelta, e ciò fornisce un risultato a mio parere controintuitivo, come si vede qui di seguito.



Dal grafico di occupazione possiamo notare, a differenza delle strategie precedenti, non ci sono livelli preferiti rispetto ad altri, come del resto c'è da aspettarsi visto che la scelta del piano è completamente casuale. Possiamo notare anche una uniformità di riempimento dei vari piani, ed è proprio questo a determinare il successo di questa strategia che pare chiaro dal grafico dei costi sottostante.



$$Utility\ Media = -2.1$$

$$Abbandoni = 0$$

I vari costi sono infatti ben distribuiti ed indipendenti dal tempo di ingresso: non c'è un trend crescente come nei grafici presentati in precedenza. Inoltre l'Utility, anche se non è la migliore finora ottenibile, rimane abbastanza alta. Questo risultato dimostra definitivamente che un fattore determinante nell'aumentare dei costi è la tendenza dei vari agenti a prendere la stesse decisioni.

Strategia Gaussiana

Questa strategia prende spunto dalle strategie a priori tentando di combinarle con la Strategia Random. Tenta infatti di risolvere il problema di omologazione delle scelte dei vari agenti attraverso la probabilità. Introduco a tal scopo:

$g = Gaussiana(0, \sigma^2)$ la varianza σ^2 varierà a seconda dei casi presi in esame.

$\Delta =$ *Incremento:* costante, anch'esso da determinare a seconda della struttura del parcheggio.

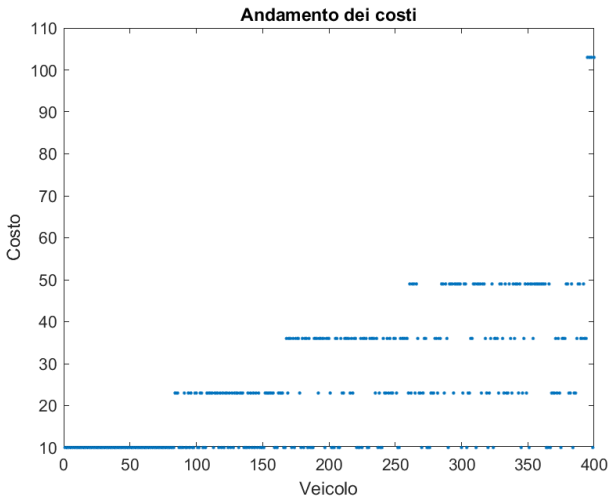
Il piano in cui effettuerò la prossima ricerca sarà:

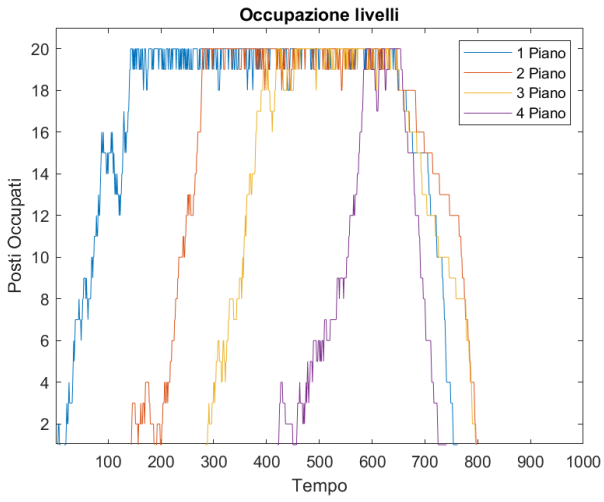
$$next\_floor = [round(|\Delta + g|)] mod(nFloors + 1) \quad (2)$$

dove $nFloors$ ricordo essere il numero totale di piani. In caso $next\_floor$ sia pari a 0, verrà impostato ad 1. In caso $next\_floor$ sia pari al piano appena visitato, verrà aumentato di una unità. Dopo diverse simulazioni, ho notato che più il numero di piani è elevato maggiori devono essere σ^2 e Δ: il loro minimo vale $\sigma^2 = 0$ e $\Delta = 1$, in questo caso la Strategia diventa completamente deterministica, ed anzi possiamo dire che "collassa" nella strategia SUSU che comunque è una delle strategie a priori migliori. A tal proposito propongo diversi confronti tra le due strategie al variare del numero di piani, mantenendo costante la capienza del parcheggio. Riporto subito che una peculiarità di questa strategia è che non ottiene alcun miglioramento dall'ibridizzazione.

Risultati Strategia Gaussiana

1) $nFloors = 4 \Rightarrow \sigma^2 = 0$ e $\Delta = 1$ Nsimulazioni=100

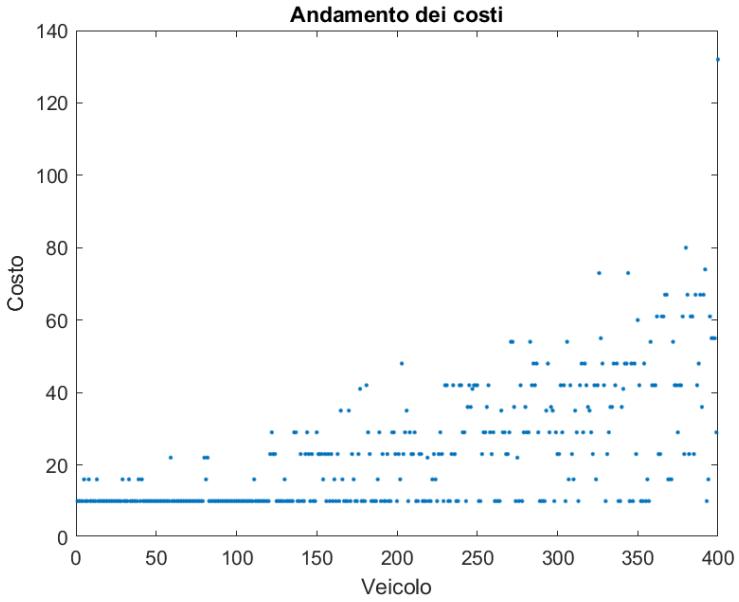




$$Utility\ Media\ Gauss = -1.89$$

$$Abbandoni = 0.2$$

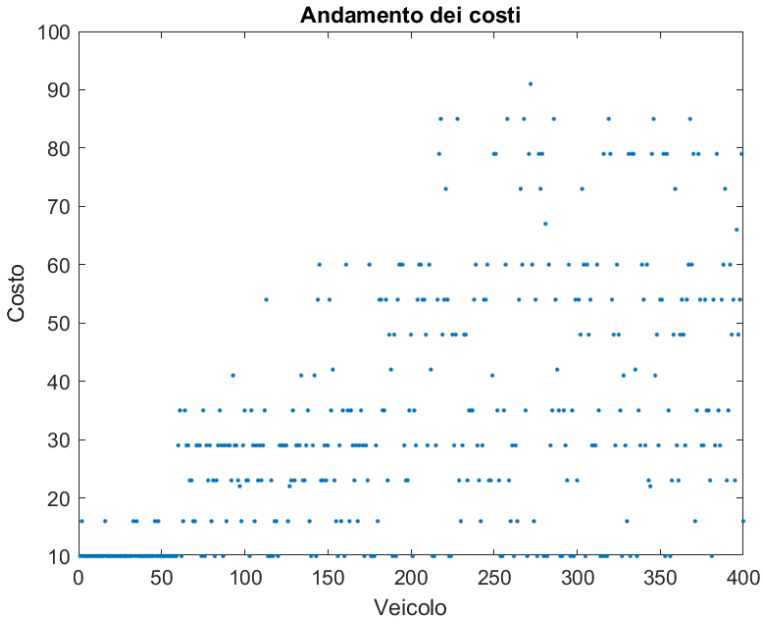
2) $nFloors = 8 \Rightarrow \sigma^2 = 0.6$ e $\Delta = L/5$ Nsimulazioni=100



$$Utility\ Media\ SUSU = -3.10$$

$$Utility\ Media\ Gauss = -3.11$$

3) $nFloors = 12 \Rightarrow \sigma^2 = 0.9$ e $\Delta = L/4$ Nsimulazioni=200



$$Utility\ Media\ SUSU = -4.51$$

$$Utility\ Media\ Gauss = -4.20$$

Possiamo notare che per $nFloors = 4$ abbiamo gli stessi risultati della strategia SUSU, lo si vede molto bene dai grafici di occupzione e di costo. Per un numero di piani basso, le due strategie sono equivalenti ma quando il numero di piani comincia ad essere elevato la strategia Gaussiana si comporta in maniera notevolmente migliore rispetto alle altre. L'unico inconveniente di questa strategia è il determinare σ^2 e Δ ottimi data la struttura del parcheggio: il metodo che ho elaborato infatti è empirico (sfrutto la simulazione), ciò non toglie che possano esistere metodi più eleganti ed efficaci per raggiungere lo stesso scopo.

Una variante di questa strategia potrebbe consistere in questa diversa politica di scelta:

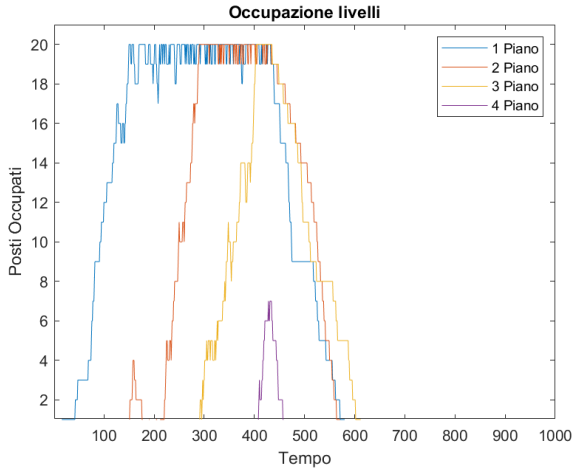
$$next\_floor = [round(|i + g|)] mod(nFloors + 1) \quad (2)$$

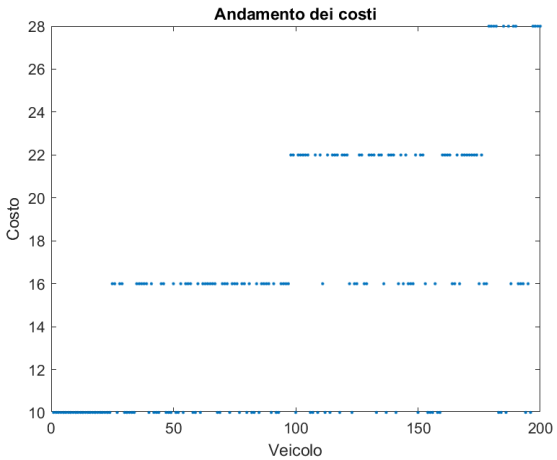
dove $i = indice\ dell'ultimo\ piano\ visitato$. Questa strategia vorrebbe simulare l'idea di ricercare "vicino" al piano appena visitato, così da evitare di spostarsi di un numero di piani troppo elevato. Tuttavia il problema che causa la sua inefficienza sono i loop di ricerca, la cui soluzione richiede una varianza elevata di g, trasformando di fatto questa strategia in una Strategia Uniforme.

Sezione V: Ulteriori Strategie

Smart Parking Strategy

Prima di analizzare altre politiche, ritengo opportuno studiare una specie di Upper Bound all'Utility ottenibile con i vincoli del modello. Per far ciò definisco questa strategia, che rappresenta un parcheggio appunto "Smart". Questo parcheggio intelligente fornisce ai propri utenti il piano più vicino disponibile. Non un'idea così futuristica, visto che tutt'ora esistono parcheggi multilivello con indicati i posti disponibili per ogni piano (ad esempio, il terminal di Roma in [3] ne è provvisto); l'unico scoglio sarebbe adattare questa tecnologia a veicoli a guida autonoma. In caso di mancanza di posti disponibili, l'agente abbandonerà direttamente la struttura con un costo pari a $Cm = 3$ dato che, in questa strategia, l'agente conosce a priori lo stato del parcheggio.





$$Utility\ Media = -1.39$$

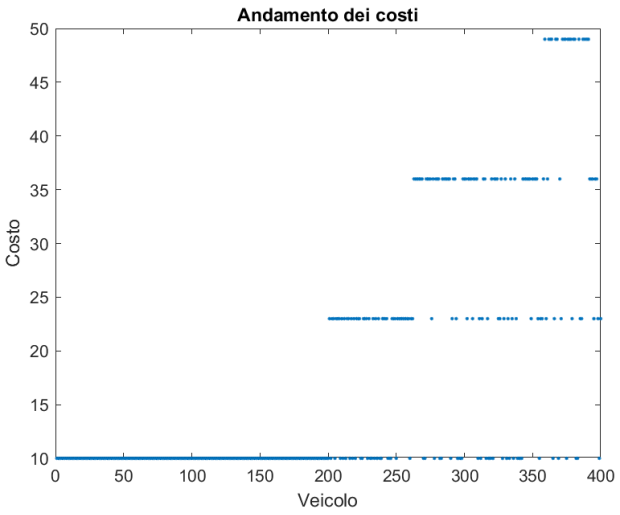
$$Abbandoni = 0$$

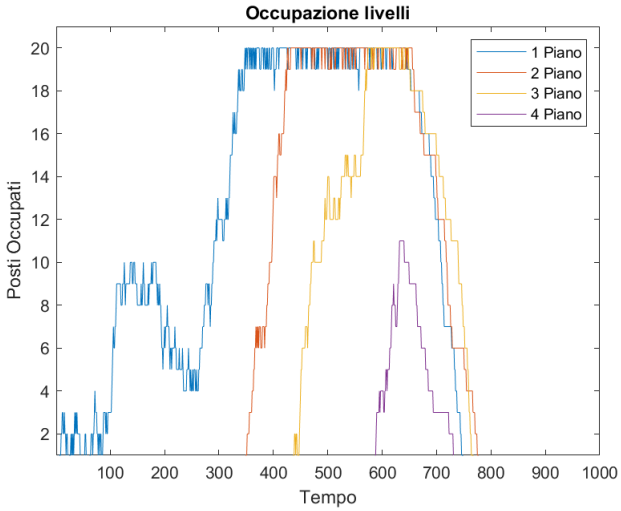
Possiamo notare dal grafico dei costi che ciascun veicolo parcheggia al primo tentativo, inoltre non essendoci abbandoni a nessuno di essi è stato attribuito un costo totale pari a 3. L'Utility è la più alta fino ad adesso, e c'era da aspettarselo visto che meglio di così non è possibile fare.

Cooperation Strategy

Dato che lo scenario è quello di veicoli a guida autonoma, è interessante studiare una strategia che preveda l'interazione tra di essi e l'ambiente circostante col fine di minimizzare il costo di parcheggio collettivo. Mentre in [20] possiamo trovare qualche spunto dal punto di vista tecnologico su come fare comunicare i diversi agenti, quello che propongo è una semplice politica, denominata Cooperation Strategy, che "combina" in qualche modo la strategia SUSU con la Smart Parking Strategy. Finora nessun veicolo sfrutta l'esperienza accumulata nel corso della ricerca di quelli precedenti, per porre rimedio a questa mancanza assumo che i diversi veicoli possano comunicare tra di loro, cosicché il veicolo i-esimo possa comunicare ai veicoli successivi gli esiti delle sue ricerche. Assumendo che i veicoli conoscano la struttura del parcheggio, ogni livello verrà etichettato come: "Full" in caso un veicolo, effettuando la ricerca, non trova posteggi liberi, mentre "Free" se riesce a parcheggiare. In caso un veicolo abbandoni la struttura, imposterà il piano come "Free". Ogni veicolo quindi scandisce il parcheggio un livello alla volta, tentando la ricerca solamente nei piani etichettati come "Free". In caso la struttura sia completamente piena, l'agente arriverà fino all'ultimo livello per poi abbandonare la struttura. Ogni veicolo però ha la possibilità di imbrogliare, con una certa probabilità. Questa scelta è dettata dal fatto che, oltre ai possibili errori che possono affliggere il sistema, nel caso il parcheggio sia a tempo, un automobilista possa imbrogliare fingendo di lasciare il parcheggio quando in realtà la macchina rimane lì. Per modellare ciò, definisco delle partenze virtuali: ad ogni partenza, ne può avvenire una di virtuale con probabilità $P_{imbroglio}$.

1) $P_{imbroglio} = 0$





$$Utility\ Media = -1.53$$

$$N.medio\ Abbandoni = 0.13$$

2) $P_{imbroglio} = 0.5 \Rightarrow Utility\ Media = -1.60 \quad N.medio\ Abbandoni = 0.15$
3) $P_{imbroglio} = 1 \Rightarrow Utility\ Media = -1.64 \quad N.medio\ Abbandoni = 0.14$

Possiamo notare quindi che all'aumentare di $P_{imbroglio}$, l'Utility Media ed il numero di abbandoni aumentano anche se non di molto, perciò un utente disonesto per ogni partenza non è sufficiente per intaccare di molto le prestazioni.

Questa Utility si colloca tra la Smart Parking Strategy e la Strategia SUSU, infatti parte dei veicoli dovrà comunque esplorare il piano col fine di segnarlo come Full eventualmente. Inoltre, a differenza della Smart Parking Strategy, i veicoli non possono abbandonare direttamente la struttura (optando invece per l'abbandono diretto, otterrei un'Utility Media più vicina alla Smart Parking Strategy, ma comunque maggiore). Ho preso questa strada perché l'informazione sullo stato del parcheggio proviene dalla cooperazione di veicoli, che possono imbrogliare. Perciò assumo che i veicoli che all'ingresso trovano l'intero parcheggio occupato comunque lo esplorino, non effettuando però ricerche di piano. I risultati sopra esposti potrebbero essere validi anche per scenari differenti in realtà. La definizione di $P_{imbroglio}$ è molto generale e può rappresentare diversi fenomeni inerenti l'occupazione abusiva di un parcheggio che dovrebbe essere libero o riservato a qualcun altro.

Cost Strategy

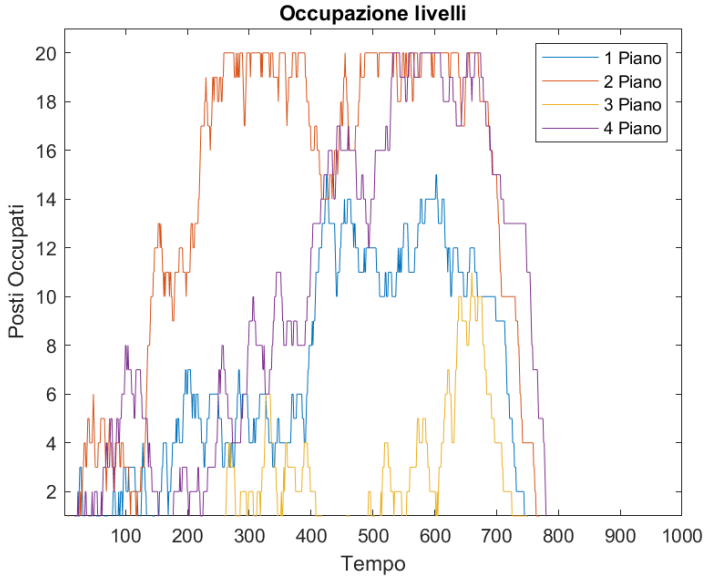
L'ultima strategia che vorrei analizzare, prende spunto da [19]: l'idea è quella di una struttura a costo variabile. Per simulare cosa succederebbe in pratica applicando questa politica, gli agenti non saranno tutti uguali, infatti saranno di tre tipi:

1) High-Cost-Agents: questi agenti danno più importanza al livello che al costo. Potrebbero essere persone con qualche problema fisico che non permette loro di fare molta strada, oppure persone (magari con buona disponibilità economica) che vogliono parcheggiare più vicino possibile.
2) Low-Cost-Agents: al cantrario del precedente, non interessa parcheggiare vicino purchè il costo del parcheggio sia minore possibile.
3) Medium-Cost-Agents: una via di mezzo dei precedenti

Per confrontare questa politica con le altre, decido di non modificare il modello di costo. La ricerca viene effettuata in questo modo: ogni veicolo può visitare solamente i livelli in cui è disposto a parcheggiare, ed in caso di esito negativo l'agente abbandonerà direttamente la struttura senza ritentare i livelli già esplorati. La struttura del parcheggio, che nelle seguenti simulazioni è costituito da quattro livelli, ha come primo livello la tariffa più costosa mentre come ultimo livello la tariffa meno costosa, i livelli due e tre hanno una tariffa variabile a seconda dei casi. La tipologia di utenti che frequentano il parcheggio segue queste probabilità:

Pr[High-Cost-Agents]=Pr[Low-Cost-Agents]=0.25, Pr[Medium -Cost-Agents]=0.5

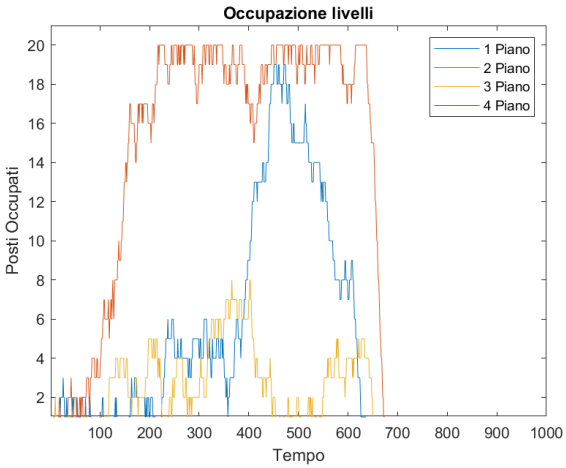
1) Livello 2 e 3: tariffa intermedia



$$Utility\ Media = -1.95$$

$$N.medio\ di\ abbandoni = 10$$

2) Livello 2: tariffa intermedia, Livello 3: tariffa bassa



$$Utility\ Media = -1.60$$

$$N.medio\ di\ abbandoni = 46.8$$

Possiamo notare che nell'ultimo caso il quarto piano non viene mai occupato ed, essendo il numero di abbandoni è molto elevato, l'Utility Media perde di significato visto che più del 10% degli utenti abbandona la struttura. Questo risultato può essere visto come una conferma di [19], in cui la prima fase di studio a priori del comportamento degli utenti non è stata fatto a dovere in quanto l'affluenza degli Medium-Cost-Agents è stata sottostimata.

Sezione VI: Conclusioni

Lo studio svolto si è occupato del confronto di Utility Functions di diverse strategie in uno scenario di Multy-Storey-Parking-Garage per veicoli a guida autonoma. Variando i paramentri di simulazione, ossia aumentando il numero di livelli e di utenti nel sistema, si è riscontrato un aumento dei costi. Le prime strategie presentate sono state quelle a priori in cui le scelte adottate dagli agenti sono deterministiche. La migliore tra di esse (in un contesto di agenti che usano la stessa politica), è la Strategia SUSU ossia una strategia che si basa della scansione sequenziale un piano alla volta. Successivamente sono state studiate le cosiddette strategie probabilistiche, che tentano di risolvere il problema delle strategie a priori che consiste nella tendenza dei veicoli ad eseguire lo stesso pattern di visita. La strategia Gaussiana, una volta impostate varianza ed incremento ottime, migliora le prestazioni della strategia SUSU in caso parcheggi con un numero di livelli sufficientemente elevato. Esistono strategie più efficienti, che però sfruttano più informazioni riguardanti il parcheggio oppure la possibilità dei veicoli di comunicare tra di loro. Infine, nell'ultimo caso, si è riscontrato un aumento dei costi nel caso i veicoli non siano completamente onesti. Di seguito riporto dei grafici che confermano i risultati appena riassunti. Per motivi di tempo di calcolo e leggibilità, i valori medi di Utility che verranno riportati saranno solamente di un numero limitato di simulazioni, 10 per la precisione. In questa maniera fornisco solamente un andamento qualitativo delle Utility, per avere stime più precise fanno fede i risultati già riportati nelle sezioni precedenti, in cui il numero di simulazioni è almeno un centinaio per ogni caso specifico.

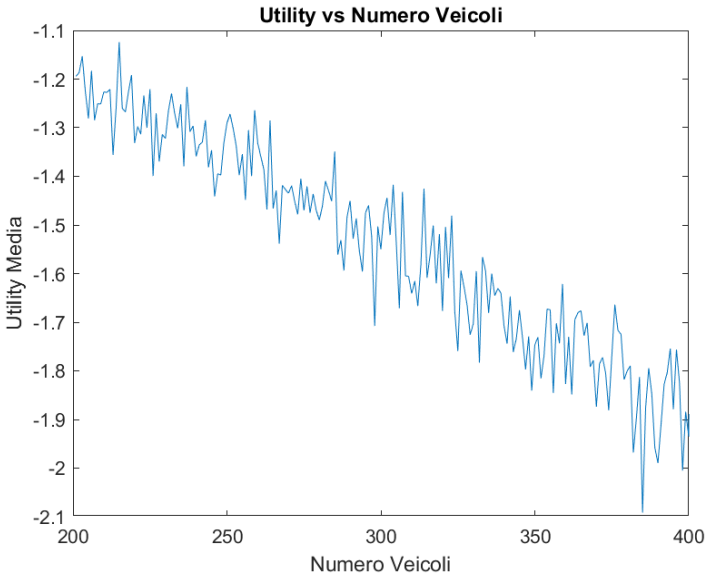


Fig.1: strategia SUSU all'aumentare del numero di veicoli

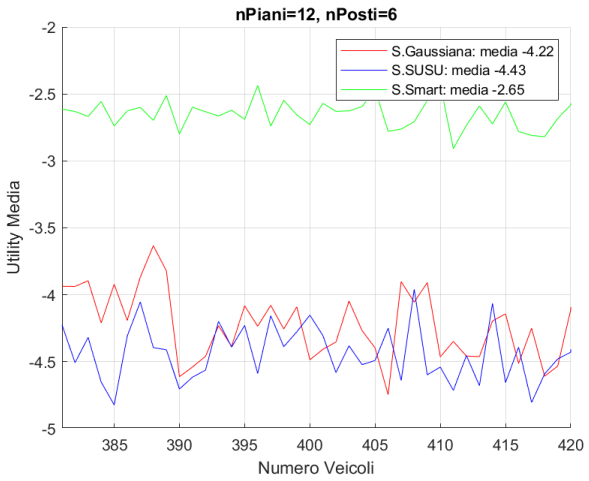


Fig.2: confronto tra Probabilistiche Vs a Priori

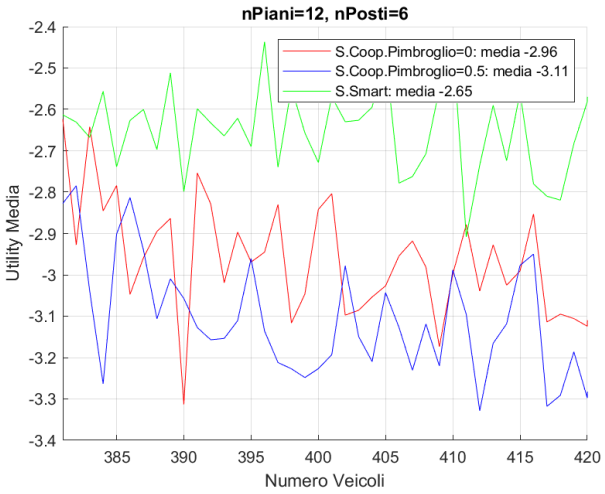


Fig.3: confronto tra veicoli onesti e non onesti

Appendice

Per verificare la correttezza del codice, e delle simulazioni, ho effettuato anche uno studio basandomi sul modello di arrivi e partenze di tipo Time-Based di [2].

La giornata viene modellata come definito in sezione II, ma la probabilità di avere arrivi nell'istante t vale:

$$P_{arrivo} = 1 - \frac{|t_{punta} - t|}{1000 - t_{punta}} \qquad (1)$$

che è massima per $t_{punta} = t$ e linearmente decrescente man mano che mi allontano da t_{punta}.

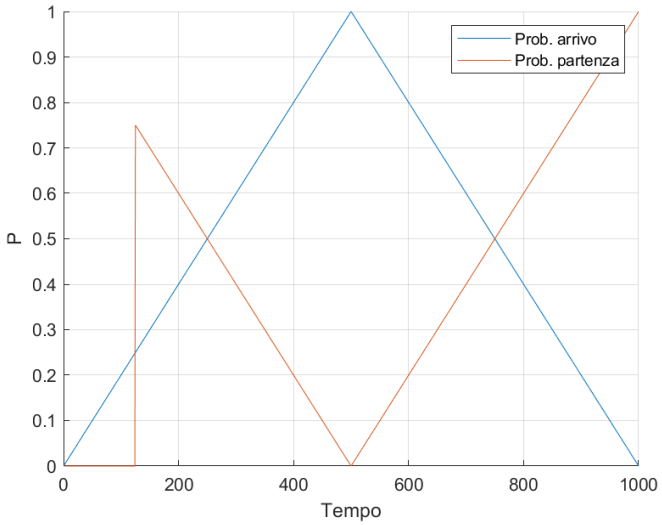
Viene definita un ulteriore probabilità $P_{partenza}$ che rappresenta la probabilità di avere una partenza nell'istante t. Più precisamente definisco $P_{partenza}$ come:

$$P_{partenza} = 1 - \frac{P_{arrivo}}{2} \text{ se } t > \frac{t_{punta}}{4}, \text{ 0 altrimenti.}$$

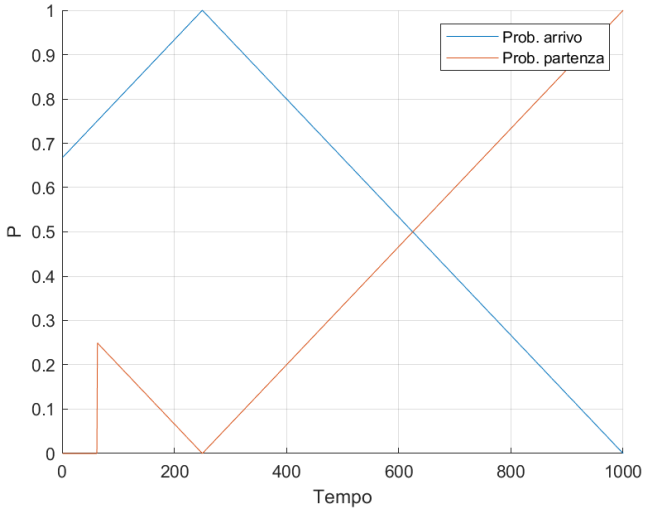
Quello che ho tentato di catturare, attraverso questa definizione, è il fatto che la probabilità che un veicolo parta è decrescente con la probabilità di arrivo, cosa non sempre vera ma valida in molti casi. Ciò dipende anche dalla fase della giornata: se l'ora di punta è nell'intorno di mezza giornata, nei primi istanti la probabilità che qualcuno parta è minima nonostante anche P_{arrivo} lo sia, dato che devono ancora arrivare veicoli. La scelta delle costanti nella definizione di $P_{partenza}$ invece è dettata dalla simulazione, infatti l'averle introdotte ha portato a un andamento degli arrivi/partenze simili a [2]. Ciò avviene perché modificare $P_{partenza}$ e P_{arrivo} influenza anche il livello di congestione del parcheggio: se in una giornata arrivano 200 veicoli in totale, ed avessi ad esempio $P_{partenza}$ non sufficientemente elevata, non potrei far fronte al numero di arrivi, perciò molti veicoli sarebbero costretti a cercare per molto tempo o abbandonare la struttura.

Di seguito riporto qualche grafico che mostra l'andamento di P_{arrivo} e $P_{partenza}$ in funzione del tempo e dell'orario di punta.

$t_{punta} = 500$



$t_{punta} = 250$



Ripetendo le stesse simulazioni, i risultati del mio modello sono coerenti con quanto riportato in [2] (anche se non direttamente confrontabili). Tuttavia ho optato per il mio modello di arrivi e partenze perché più vicino alla realtà.

Breve discussione sul codice

Come accennato in precedenza, il simulatore consiste in un codice MatLab. Ho optato per una programmazione ad oggetti definendo le classi:

- AgentDefault: oggetto vettura, che ha come campi:
  - name: "Default" se non diversamente impostato
  - prevFloor: piano precedentemente visitato
  - currentFloor: piano corrente
  - P: importanza al costo del parcheggio (0=per nulla importante, 1=abbastanza importante, 2=importantissimo)
  - currentTimeCost: costo associato alla ricerca all'istante attuale
  - trovato: variabile booleana che indica lo stato di ricerca
  - politica: strategia di ricerca

- Piano: classe utile per la gestione dei piani
    - freeSpots: indica il numero di posti liberi
    - status: indica lo stato del piano, può valere “Full” o “Free”
    - tariffa: può essere 0,1 o 2 a seconda che il costo associato a parcheggio nel piano sia costosto, normale o economico rispettivamente.

Infine è presente la classe PoissonSimulation che consiste nel main del codice. Per gestire la parte aleatoria del programma, ho fatto uso delle funzioni randi(·) e binornd(·,·) messe a disposizione dalla suite di MatLab. Per ulteriori dettagli, il codice è stato commentato in modo tale da facilitarne la comprensione. Di seguito riporto il link dal quale è possibile scaricare il programma:

https://drive.google.com/drive/folders/1ZfUcI-8Z4Sp5ENPODR2F8zZ8N-xUR3ep?usp=sharing

Riferimenti

[1] P. Zhao and M. Zhang, “The impact of urbanisation on energy consumption: A 30-year review in China”, *Urban climate*, vol. 24, pp. 940-953, 2018.

[2] E. Gindullina, S. Mortag, M. Dudin and L. Badia, “Multi-Agent Navigation of a Multi-Storey Parking Garage via Game Theory*”, IEEE 22nd Int. Sym. on a World of Wir. Mob. and Mul. Net. (WoWMoM)*, pp. 280-285, 2021.

[3] Aeroporti di Roma, [online] Available: https://www.adr.it/parcheggi-terminal-a-b-c-d#

[4] A. Cappiello, I. Chabini, E. K. Nam, A. Lue and M. Abou Zeid, "A statistical model of vehicle emissions and fuel consumption," *Proc. IEEE 5th Int. Con. on Int. Transp. Sys.,* pp. 801-809, 2002.

[5] I. Barabàs, A. Todoruţ, N. Cordos, and A. Molea, “Current challenges in autonomous driving”, *IOP Conf. Ser. Mat. Scie. and Eng.,* vol. 252., 2017.

[6] W. Yan-ling, W. Xin and Z. Ming-chun, “Current situation and analysis of parking problem in Beijing”, *Proc. eng*, vol.137, pp.777-785, 2016.

[7] M. Campbell and M. Egerstedt, “Autonomous driving in urban environments: approaches, lessons and challenges”, *Phil. Tran. of the Royal Soc. A: Math., Phy.and Eng. Scie.,* vol. 368, no. 1928, pp. 4649-4672, 2010.

[8] T. Shen, Y. Hong, M. M. Thompson, J. Liu, X. Huo and W. Lian, “How does parking availability interplay with the land use and affect traffic congestion in urban areas? The case study of Xi’an, China”, *Sustainable Cities and Society,* vol. 57, pp. 102126, 2020.

[9] M. Gallo, L. D'Acierno and B. Montella, “A multilayer model to simulate cruising for parking in urban areas”, *Transport policy*, vol.18, no. 5, pp. 735-744, 2011.

[10] B. Dogaroglu, S. P. Caliskanelli and S. Tanyel, “Comparison of Intelligent Parking Guidance System and Conventional System with Regard to Capacity Utilisation”, *Sus. Cit. and Soc*., vol. 74, pp. 103152, 2021.

[11] J. Hanzl, “Parking information guidance systems and smart technologies application used in urban areas and multi-storey car parks”, *Trans. Res. Proc*., vol. 44, pp. 361-368, 2020.

[12] W. Alsafery, B. Alturki, S. Reiff-Marganiec and K. Jambi, "Smart Car Parking System Solution for the Internet of Things in Smart Cities," *1st Int. Conf. on Comp. App. & Info. Sec. (ICCAIS)*, pp. 1-5, 2018.

[13] J. Polak and K. W., Axhausen, “Parking search behaviour: A review of current research and future prospects”, *Trans. Stud. Units, Wor. Pap.,* vol. 540, 1990.

[14] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta and J. Lin, “Parking slot assignment games”. *In Proc. of the 19th ACM SIGSPATIAL Inter. Conf. on Adv. in Geo. Info. Sys. (GIS '11). Ass. for Comp. Mac.,* pp. 299–308, 2011.

[15] A. John, “An overview of the energy efficiency potential”, *Env. Inn. and Soc. Trans.,* vol. 9, pp. 38-42, 2013.

[16] C.C. Mendat and M.S. Wogalter, “Perceptions of Parking Facilities: Factors to Consider in Design and Maintenance”, *Proc. of the Hum. Fac. and Erg. Soc*, vol. 47, no. 6, pp. 918-921, 2003.

[17] A. V. Guglielmi and L. Badia, “Analysis of strategic security through game theory for mobile social networks,” *IEEE Int. Comp. Aided Model. Design Commun. Links and Networks (CAMAD)*, pp. 1-6, 2017.

[18] T. Vanderbilt and B. Brenner, “Traffic: Why We Drive the Way We Do (and What It Says about Us)”, 2010.

[19] U. Michieli and L. Badia, “Game theoretic analysis of road user safety scenarios involving autonomous vehicles," *Proc. Ann. Int. Symp. Pers., Indoor Mob. Radio Commun. (PIMRC)*, pp. 1377-1381. IEEE, 2018.

[20] R. Nagel, S. Eichler and J. Eberspacher, “Intelligent wireless communication for future autonomous and cognitive automobiles”, *IEEE Int. Vehicle. Symp*., pp. 716-721, 2007.

[21] D. Teodorović and P. Lučić, “Intelligent parking systems”, *Eur. Jour. of Op. Res*., vol. 175, no. 3, pp. 1666-1681, 2006.

[22] L. X. Wang and J. M. Mendel, “Generating fuzzy rules by learning from examples”, *IEEE Trans. on sys., man, and cyb*., vol. 22, no. 6, pp. 1414-1427, 1992.

[23] Y. Wang, M. Li, X. Li and F. He, “Online operations strategies for automated multistory parking facilities”, *Trans. Res. Part E: Log. and Trans. Rev*., vol. 145, pp. 102135, 2021.

[24] A. V. Guglielmi and L. Badia, “Bayesian game analysis of a queueing system with multiple candidate servers,” *IEEE Int. Comp. Aided Model. Design Commun. Links and Networks (CAMAD)*, pp. 85-90, 2015.

[25] G. Maternini, F. Ferrari and A. Guga, “Application of variable parking pricing techniques to innovate parking strategies. The case study of Brescia”, *Case stud. on trans. pol*., vol. 5, no.2, pp. 425-437, 2017.

[26] L. Badia and M. Zorzi, “Dynamic utility and price based radio resource management for rate adaptive traffic,” *Wireless Networks*, vol. 14, no. 6, pp.803-814, 2008.

[27] P. L. Krapivsky and S. Redner, “Simple parking strategies”, *Jou. of Stat. Mech.: The. and Exp*, vol. 2019, no. 9, pp. 093404, 2019.

[28] A. V. Guglielmi and L. Badia, “Social communication to improve group recognition in mobile networks,” In *IEEE Globecom Workshops (GC Wkshps)*, pp. 1-6, 2015.

[29] L. Badia, “Evaluation of a potential energy methodology for joint routing and scheduling in wireless mesh networks,” In *6th Int. Conf. Mobile Adhoc Sensor Syst.*, pp. 657-662, 2009.

[30] S. Azad, L. Badia, A. Rahman, and J.M. Zain, “Raising fairness issue of vehicle routing problem,” *IEEE Int. Conf. Intell. Transp. Engin. (ICITE)*, pp. 13-17, 2016.