ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA GESTIONALE

TESI DI LAUREA

in

Resource Optimization

SVILUPPO DI UN MODELLO DI OTTIMIZZAZIONE PER IL CAR SHARING ELETTRICO DELL'UNIVERSITÀ DI BOLOGNA

CANDIDATO

Francesco Maria Marchese

RELATORE

Chiar.mo Prof.
Daniele Vigo

CORRELATORI

Prof. Enrico Malaguti

Ing. Filippo Masini

Anno Accademico 2014/2015

Sessione I

A te e alla mia famiglia

Indice

Introduzione

In occasione del tirocinio curriculare ho scelto di partecipare al progetto di ricerca europeo denominato "E4-share" (*Models for Ecological, Economical, Efficient and Electric* car sharing), un'iniziativa volta ad esplorare soluzioni innovative per l'organizzazione di un servizio di car sharing urbano con veicoli elettrici. L'obiettivo del mio lavoro è stato quello di fornire supporto al Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione nell'analisi del car sharing elettrico dell'Università di Bologna, riservato al proprio personale docente e tecnico-amministrativo.

Nel corso di questa esperienza ho avuto l'opportunità di affrontare in prima persona le problematiche legate alla pianificazione di una tipologia di servizio che rappresenta una nuova sfida in termini di sostenibilità ambientale e razionalizzazione degli spazi pubblici. La mia attività è stata prevalentemente finalizzata all'acquisizione di dati critici sul car sharing universitario, al fine di consentire l'implementazione di nuovi metodi numerici, esatti ed euristici, in grado di migliorarne la gestione operativa. L'argomento della tesi nasce dunque dalla volontà di sviluppare un modello di ottimizzazione per incrementare l'efficienza del servizio, il cui obiettivo principale è soddisfare le richieste degli utenti attraverso un corretto utilizzo della flotta a disposizione.

Il primo capitolo affronta il ruolo del car sharing nel trasporto urbano come strumento in grado di modificare gli attuali modelli comportamentali nell'uso dell'automobile e di favorire una mobilità più razionale, economica ed ecocompatibile. La strada rimane tuttora il regno incontrastato delle auto tradizionali a combustione interna, ma il preoccupante scenario ambientale ha ridestato l'attenzione verso i veicoli elettrici. La cosiddetta "ansia da autonomia", dovuta alla limitata capacità di stoccaggio di energia delle batterie, e la mancanza di un'adeguata infrastruttura di ricarica restano tuttavia gli ostacoli principali alla loro diffusione. Nella prima parte del lavoro vengono dunque discussi i motivi che incentivano l'integrazione delle auto elettriche nel car sharing, sebbene essi introducano maggiori difficoltà alla gestione del servizio. Successivamente, un'analisi delle componenti del car sharing consente di rappresentare, attraverso i diagrammi delle classi e delle attività, la sua struttura complessiva, gli elementi distintivi, le relazioni tra essi ed i processi che lo caratterizzano.

Il secondo capitolo descrive l'organizzazione del servizio di condivisione di auto elettriche dell'Università di Bologna, con particolare riferimento alla flotta impiegata, ai luoghi nei quali

sono aggregati i veicoli (denominati isole), alle stazioni di ricarica, agli utenti e al processo di gestione delle prenotazioni. Inoltre, dalla consultazione dei dati dei viaggi effettuati dal personale strutturato universitario, sono state ricavate le istanze di prova per un problema di schedulazione di veicoli con limitata autonomia.

Quest'ultimo viene definito e risolto nel terzo capitolo attraverso lo sviluppo in linguaggio MATLAB di un algoritmo *greedy*, il cui obiettivo è minimizzare il numero di auto dell'Università necessarie a soddisfare la domanda giornaliera, considerando come vincoli gli orari di prenotazione dei viaggi e i chilometri di autonomia delle vetture. Il risultato dell'euristica è una schedulazione ammissibile per ciascun veicolo utilizzato, ovvero una sequenza di viaggi che soddisfa i limiti imposti.

Il car sharing dell'Università di Bologna comprende quaranta auto elettriche ed i potenziali utenti sono circa seimila, dunque è necessario modellare un problema di distribuzione giornaliera della flotta nelle isole, utilizzando un minor numero di risorse rispetto a quante ne sono richieste. Le auto vengono impiegate diversamente in termini di numero di utenti serviti, durata dei viaggi effettuati e chilometri percorsi, pertanto, sulla base della soluzione dell'algoritmo, è possibile associare ad ogni veicolo un costo.

Nel quarto ed ultimo capitolo, dopo una breve descrizione preliminare del metodo del *Branch and Bound*, è proposto un modello di ottimizzazione orientato alla minimizzazione di due voci di costo: il mancato servizio associato all'impossibilità di soddisfare tutta la domanda e la movimentazione delle auto da una stazione all'altra. Il modello è sviluppato in linguaggio MATLAB e la sua soluzione è ricavata utilizzando uno dei solutori inclusi nell'*Optimization Toolbox* di MATLAB.

Capitolo 1
Il car sharing elettrico

1.1 *Il fenomeno del car sharing*

Come ogni anno, l'Istat ha recentemente aggiornato il paniere di riferimento per il calcolo dell'inflazione e a rappresentare le nuove abitudini di consumo delle famiglie italiane c'è l'ingresso del car sharing. Il dato significativo che emerge da questa decisione è che nel nostro Paese sta aumentando la consapevolezza della necessità di affermare una nuova visione di mobilità urbana. Secondo il Presidente della Fondazione per lo sviluppo sostenibile, il servizio di condivisione delle auto rappresenta uno dei settori più promettenti e vitali della *green economy*. Utilizzare il car sharing vuol dire ridurre le emissioni inquinanti, razionalizzare gli spazi pubblici, risparmiare sui costi di proprietà dell'auto privata e soprattutto garantire ai cittadini una migliore qualità della vita. Il servizio, nato in Svizzera nel 1948 per iniziativa di alcuni privati motivati da ideali ecologici, consente agli utenti di disporre di una flotta comune di veicoli distribuiti su più aree di parcheggio, localizzate in prossimità di importanti nodi di scambio intermodale. L'accesso alle auto avviene in modo autonomo ed il costo complessivo sostenuto dall'utente comprende in genere una quota fissa d'iscrizione e una tariffa proporzionale al tempo di utilizzo e alla percorrenza realizzata.

La condizione favorevole di sviluppo del servizio risiede principalmente nell'attuale rigidità del mercato che offre ampie possibilità di scelta a chi desidera acquistare un veicolo, ma concede poche alternative, economiche e funzionali, a chi ne fa un uso occasionale. Il car sharing si rivolge proprio a quest'ultima categoria di automobilisti perché offre la possibilità di muoversi senza sostenere i disagi e i costi fissi legati al possesso dell'auto. Inoltre, allenta la morsa del traffico veicolare nei centri urbani e favorisce comportamenti individuali più razionali nell'uso della macchina, a vantaggio di mezzi ecocompatibili e a basso impatto ambientale.

Un dato estremamente rilevante è il fatto che l'80% delle auto private circolanti in Europa viaggi in media per circa un'ora, occupando spazi di sosta per il resto del tempo. A differenza del *car pooling* che prevede l'utilizzo dell'autovettura attraverso un accordo preliminare tra persone che compiono lo stesso percorso nello stesso periodo, la forza della condivisione nel car sharing risiede nella possibilità per gli utenti di utilizzare autonomamente e in periodi

diversi lo stesso veicolo. La configurazione del servizio non è nemmeno assimilabile alle forme tradizionali di autonoleggio che rappresentano la soluzione ideale per spostamenti lunghi e protratti nel tempo. Il vantaggio economico dipende comunque dalle tariffe applicate e si riduce, fino ad annullarsi, al crescere delle percorrenze realizzate; la soglia di convenienza in media si assesta intorno ai diecimila chilometri annui.

La riduzione dei volumi di traffico e dello spazio necessario per la sosta costituiscono due importanti benefici dell'applicazione del servizio, infatti è stato osservato che ogni auto condivisa possa sostituire fino a sedici auto private. L'espansione del servizio su scala più ampia non può comunque prescindere dall'esistenza di un efficace sistema di trasporto pubblico, in quanto il car sharing è un servizio complementare ad esso e non sostitutivo.

La collocazione delle aree di parcheggio costituisce un pilastro nell'organizzazione del sistema, poiché una corretta progettazione permette di ridurre la percorrenza chilometrica media, il traffico e l'impatto delle emissioni atmosferiche. Per questa ragione, l'individuazione delle aree di sosta è il risultato di un'analisi approfondita del territorio cittadino che prende in considerazione indicatori socio-economici e di trasporto in riferimento alla mobilità degli abitanti.

Come accade in tutti i settori della produzione, uno dei problemi fondamentali del car sharing è rappresentato dal dimensionamento ottimale del servizio: il gestore deve infatti operare con adeguati livelli di convenienza economica.

Dal punto di vista dell'utente, il successo del sistema dipende dalla disponibilità del veicolo nel momento in cui è richiesto, viceversa per l'operatore è prioritario l'obiettivo di soddisfare le esigenze degli utenti con un numero minimo di veicoli. Si può dunque sostenere che tra i più importanti parametri di valutazione dell'efficacia e dell'efficienza del servizio figurano il numero medio di utenti per auto e il tasso di utilizzo dei veicoli.

Le organizzazioni europee di maggior successo operano mediamente con un veicolo ogni venti o trenta persone, ma l'esperienza dimostra che questo numero dipende dalle dimensioni del servizio stesso. Sulla base di alcune indagini svolte presso le principali compagnie di car sharing, risulta che il cliente medio ha un'età compresa tra 25 e 34 anni nel 41% dei casi (fino a 50 anni nel 70%), un impiego a tempo pieno ed un livello di istruzione medio-alto. Tra le motivazioni più importanti per l'adesione al servizio figurano la vicinanza all'area di parcheggio, l'elevata disponibilità del veicolo, le basse tariffe di utilizzo, l'affidabilità dei veicoli e la flessibilità nelle prenotazioni.

Oltre ai vantaggi economico-sociali apportati dalla condivisione dell'auto, è opportuno considerare il contesto nel quale si colloca questo fenomeno. Il tema del futuro dei mercati viene affrontato frequentemente, ma nella quasi totalità dei casi non ci si sofferma a riflettere sui cambiamenti che potrebbero modificare radicalmente la nostra vita e sull'impatto che essi potrebbero avere sull'economia globale.

Un megatrend è un cambiamento strutturale di lungo periodo che impatta sulle dinamiche sociali, economiche e culturali, dunque studiarne gli effetti consente di individuare dei settori che con il tempo possono costituire un'interessante alternativa in termini di diversificazione. In tal senso, secondo il settimanale *Time*, il 2015 sarà l'anno della definitiva esplosione della cosiddetta *sharing economy*, ovvero del sistema socio-economico basato sul consumo collaborativo, all'interno del quale la mobilità ha un ruolo estremamente rilevante.

Da una logica incentrata sul possesso si passa ad un'altra basata sull'accesso perché la condivisione, attraverso l'utilizzo delle tecnologie, risulta più efficiente, sostenibile ed aiuta a costruire una comunità. Molti sono i settori e le realtà da citare come esempi: *BlaBlaCar* consente di condividere un passaggio in auto, *Okobici* le biciclette private, *Fubles* per partecipare ad una partita di calcio, *swap.com* per lo scambio di oggetti e *Gnammo* per organizzare eventi culinari.

In Italia il 13% della popolazione ha utilizzato almeno una volta le opportunità offerte dalla *sharing economy* e, nonostante la differenza con i numeri di Stati Uniti (52%), Gran Bretagna (64%) e Francia (50%), anche nel nostro Paese sta aumentando la consapevolezza dei vantaggi della condivisione di beni e servizi. Questo cambiamento sta dunque rivoluzionando la cultura del consumo e, di conseguenza, anche le logiche della produzione.

La crisi economica ha chiaramente avuto un impatto determinante sulla progressiva diffusione di modalità di trasporto alternative. Tra il 2008 ed il 2013 si è riscontrato un crollo del 22% della mobilità (uno spostamento su cinque non viene più effettuato), riconducibile alle difficoltà del Paese.

Un altro processo legato alla crisi è il progressivo invecchiamento del parco veicolare circolante, infatti se nel 2007 l'età media delle vetture era di circa sette anni e mezzo, nel 2013 è salita a nove anni e nove mesi, causando maggiore inquinamento e minore sicurezza. Il car sharing è dunque un servizio che presuppone e insieme sollecita un profondo cambiamento culturale dei cittadini nelle scelte di mobilità perché scinde l'utilizzo del veicolo dal possesso individuale e ne ottimizza l'impiego attraverso un meccanismo di rotazione degli utenti.

Molte case automobilistiche, i cui interessi economici si scontrano con la politica della condivisione (nonostante la collaborazione strategica con gli operatori del servizio rappresenti un efficace strumento di marketing), sono convinte che il car sharing non costituisca un elemento strutturale rilevante nel settore dell'*automotive* perché nel mercato europeo sono vendute ogni anno più di dieci milioni di auto. Tuttavia, se da un lato è vero che la quota di espansione del servizio è ancora marginale rispetto agli acquisti, studi del settore prevedono per i prossimi anni una crescita esponenziale della dimensione della flotta e del numero di utenti: nel 2020 ci saranno 450.000 vetture utilizzate da circa 25 milioni di membri, a fronte delle 70.000 auto e dei 3,5 milioni di utenti del 2013. Il Nord America e l'Europa sono le aree del mondo nelle quali il servizio è maggiormente diffuso, attraverso le offerte dei principali operatori del mercato: Zipcar ed Enterprise negli Stati Uniti, Car2Go e Drivenow in Germania, Car2Go ed Enjoy in Italia. Nel panorama attuale possiamo distinguere quattro diversi modelli di car sharing:

− *two-way*. Gestito da amministrazioni pubbliche in molte città, prevede che il veicolo, al termine del viaggio, venga riportato nella stessa stazione di prelievo;

− *one-way*. Consente di restituire l'auto in un altro parcheggio riservato al servizio, con conseguente necessità di ricollocamento della flotta per garantire l'incontro tra domanda ed offerta in ciascuna stazione;

− *free floating*. Di fatto ha rivoluzionato e diffuso l'utilizzo del car sharing perché dà la possibilità di prenotare l'auto (localizzata presso parcheggi pubblici a pagamento o meno) attraverso il web o un'applicazione e di lasciarla ovunque si desideri all'interno dell'area coperta dal servizio;

− *peer to peer*. A differenza degli altri modelli che sono *business to consumer*, l'ultimo è di tipo *consumer to consumer* perché consente agli utenti di noleggiare la propria auto a terzi tramite un operatore che fa da intermediario tra domanda e offerta. Il car sharing *peer to peer* in Italia non esiste perché ci sono problemi burocratici legati alla copertura assicurativa delle auto private per uso commerciale.

La progressiva riduzione del tasso di motorizzazione e l'aumento della diffusione di veicoli a basso impatto ambientale (a metano, gpl, a trazione ibrida ed elettrica), i quali complessivamente raggiungono l'8% del parco nazionale circolante, rappresentano evidenti segnali di sviluppo di una mobilità sempre più sostenibile. Nell'ultimo anno il car sharing ha registrato sul territorio nazionale un aumento degli iscritti del 7,8%, ma, nonostante la crescita della domanda, i veicoli condivisi sono inspiegabilmente diminuiti del 4,5%. Il nostro Paese

deve dunque continuare a sostenere l'eco-mobilità attraverso iniziative di condivisione delle auto, per le quali sono stati ottenuti anche importanti riconoscimenti: nel 2014 Milano è stata premiata come migliore città europea per il car sharing *free floating*.

1.2 *Le auto elettriche*

La nascita dell'auto elettrica risale al XIX secolo, quando, tra il 1832 ed il 1839, lo scozzese Robert Anderson inventò la prima vettura di questo tipo, la cui origine è pertanto antecedente rispetto a quella a combustione interna. La Francia e la Gran Bretagna furono le prime nazioni europee ad avere una diffusione significativa di veicoli elettrici a partire dalla fine del XIX secolo, ma la prima applicazione commerciale su larga scala si ebbe negli Stati Uniti nel 1897 quando l'intera flotta dei taxi di New York divenne elettrica.

Le automobili erano inizialmente disponibili nelle versioni elettrica, a vapore e a benzina, ma le prime erano preferite rispetto alle altre perché erano silenziose, non vibravano, non emettevano sgradevoli odori e non avevano il cambio. Così tra il 1899 e il 1900, negli Stati Uniti le auto di tipo elettrico furono in assoluto le più vendute, mentre quelle a vapore furono presto abbandonate perché richiedevano lunghi tempi di avviamento (fino a 45 minuti).

Ma a partire dal 1920, una serie di fattori economico-ambientali segnarono un cambio di tendenza a favore dell'auto a benzina, eclissando il successo dell'auto elettrica. Innanzitutto la scoperta di grandi giacimenti di petrolio in Texas diminuì drasticamente il prezzo di mercato del carburante, poi l'introduzione di importanti innovazioni, come la catena di montaggio dell'industriale Henry Ford e il motorino di avviamento, segnarono una svolta decisiva della tendenza. Queste nuove invenzioni permisero infatti di ridurre i costi di produzione e di rendere i prezzi delle auto accessibili a una maggiore fetta di mercato.

Giocò un ruolo importante anche la costruzione di infrastrutture stradali che collegavano centri molto distanti tra loro, il che rendeva evidente il problema legato all'autonomia delle auto elettriche. Tra la fine degli anni Sessanta e i primi anni Settanta ci fu un nuovo interesse verso le auto elettriche, a fronte della nascente attenzione al miglioramento della qualità dell'aria e del vertiginoso innalzamento del prezzo del petrolio. Quest'ultimo raggiunse un livello fino a tre volte superiore del prezzo originario, a seguito della guerra arabo-israeliana dello Yom Kippur, culminata nell'embargo petrolifero dei paesi OPEC (*Organization of the Petroleum Exporting Countries*) a scapito degli USA e di altri Paesi occidentali. Tuttavia, questa nuova fase non portò ad una vera e propria diffusione dei veicoli elettrici.

Negli ultimi anni un aumento costante del livello di inquinamento ha ridestato l'interesse dei governi e dell'industria verso un nuovo concetto di mobilità ed una maggiore sensibilità verso un futuro più pulito. A quattro anni dall'avvio, si è appena concluso con successo il progetto europeo Green eMotion che ha definito e collaudato un quadro di riferimento per gli operatori della mobilità elettrica, con l'obiettivo di realizzare un ecosistema integrato, efficiente e interoperabile. Questa importante iniziativa ha definito l'architettura informatica e di rete europea necessaria ad assicurare un collegamento adeguato fra tutti gli operatori di mercato, consentendo agli utilizzatori di veicoli elettrici di avere facile accesso all'infrastruttura di ricarica.

Un requisito necessario per lo sviluppo e la diffusione della mobilità elettrica è l'accettazione sociale, tuttavia serve anche un chiaro impegno politico, oltre che dell'industria e dei servizi, a favore di un sistema orientato ai consumatori, attraverso la promozione di offerte studiate per rispondere alle loro esigenze.

Negli scenari di decarbonizzazione dell'economia elaborati dalla Commissione Europea nella *Energy Roadmap* per il 2050, l'energia elettrica giungerà ad un tasso di penetrazione pari al 36-39% della domanda energetica complessiva, con quasi un conseguente raddoppio della sua importanza rispetto al 20-21% di oggi. La sua grande versatilità d'impiego, la facilità con cui può essere trasportata sulle reti di distribuzione e trasmissione, i nuovi impieghi che si affermano sempre di più su scala industriale e l'elevata incidenza delle fonti di energia rinnovabile, sono alla base di tali proiezioni.

Tra i nuovi utilizzi, un ruolo di primaria importanza è fornito dalla mobilità, poiché la stessa Commissione stima che nel 2050 il 65% della domanda di autovetture e veicoli leggeri sarà indirizzata verso motori elettrici. Questi ultimi sfruttano l'energia proveniente da batterie (le auto vengono indicate anche con l'acronimo BEV, *Battery Electric Vehicle*), le quali vengono ricaricate per il rifornimento attraverso l'elettricità proveniente dalla rete; la sostituzione rapida delle batterie in un'apposita stazione automatizzata è un'innovativa alternativa alla ricarica ed ai suoi lunghi tempi. Le auto elettriche hanno un notevole rendimento, decisamente superiore a quello delle vetture a combustione interna: il 90% dell'energia potenziale si trasforma in energia meccanica, mentre nei motori a benzina e diesel soltanto il 25% ed il 40% rispettivamente, il restante si disperde in calore.

In uno scenario attuale nel quale il 12% delle emissioni di anidride carbonica in Europa è dovuto al traffico automobilistico, la caratteristica più rilevante nell'utilizzo di un'auto elettrica è certamente la mancanza di emissioni nocive. Non è possibile considerarla un mezzo a impatto

zero perché l'elettricità attualmente prodotta proviene per la maggior parte da combustibili fossili, ma negli ultimi anni vi è stata una crescita straordinaria delle nuove installazioni di impianti a fonti rinnovabili, le quali incidono per più di un quarto del totale dei consumi elettrici.

Mobilità elettrica e produzione di energia rinnovabile, in particolare fotovoltaica ed eolica, costituiscono dunque un binomio che, se opportunamente coordinato, può migliorare la qualità dell'aria nelle città e ridurre drasticamente il costo della mobilità urbana dalle quindici alle venti volte rispetto a quello attuale. Un problema di tipo ecologico è rappresentato dalle batterie al litio che, sia in caso di smaltimento che di rigenerazione (al termine del loro ciclo vitale possono essere rigenerate fino all'80% della loro capacità iniziale), possono creare diversi problemi a livello ambientale. Un ulteriore grande vantaggio dei veicoli elettrici è la loro silenziosità che permette una drastica riduzione dell'inquinamento acustico. Sul fronte delle spese, esse comportano una minore manutenzione (niente cambio dell'olio, controlli anti-inquinamento non necessari, sostituzione liquidi ridotta, parti meccaniche più facilmente riparabili), ma, per contro, ci sono da registrare gli elevati costi di acquisto e di sostituzione delle batterie. Decisamente interessante, invece, è la convenienza energetica perché percorrere cento chilometri costa circa tre euro.

1.3 *L'integrazione funzionale tra car sharing e auto elettriche*

I livelli preoccupanti dell'inquinamento atmosferico e le notizie sullo sfruttamento delle risorse petrolifere riaprono periodicamente il dibattito su impatto ambientale e convenienza dei mezzi comunemente alimentati a combustibili fossili. Quasi il 70% delle emissioni totali di CO2 nel settore dei trasporti viene prodotto su distanze inferiori ai cinquanta chilometri.

La ricerca di un'alternativa alle auto tradizionali finisce spesso per indicare i veicoli elettrici come possibile soluzione ai problemi della mobilità urbana, in quanto l'elettricità può essere generata da fonti ecosostenibili e le auto non emettono gas di scarico inquinanti quando vengono utilizzate.

Tuttavia l'auto elettrica non ha mai sostituito quella tradizionale a causa di due limiti tecnici essenziali: la quantità di energia immagazzinabile dalle batterie e il loro tempo di ricarica (limiti che non riguardano solo l'auto ma tutti i dispositivi elettronici). La maggior parte dei modelli venduti non supera i 160 chilometri reali (l'autonomia dei veicoli tradizionali è 6-7 volte

superiore), i tempi di ricarica si misurano in ore e le batterie sono molto costose perché impiegano metalli come il litio, il nichel ed il cadmio che sono disponibili in quantità limitata.

Bisogna comunque considerare che si stanno sempre più diffondendo sistemi di ricarica rapida che consentono di rifornire completamente l'auto in appena trenta minuti. L'utilizzo di veicoli elettrici nel car sharing rappresenta un'efficace strategia per minimizzare l'impatto di tali vincoli energetici, poiché le distanze medie percorse dagli utenti sono compatibili con l'autonomia disponibile.

La condivisione dell'auto può rappresentare un'opportunità per la diffusione dei veicoli elettrici e viceversa, andando così a costituire un rivoluzionario paradigma della mobilità che combina innovazione tecnologica e di servizio: l'*e*-car sharing. Questo modello va nella direzione opposta rispetto a quello attualmente più diffuso che prevede ancora l'utilizzo di auto private con motore a combustione interna, alimentato a benzina, gasolio o gas.

Queste due soluzioni, grazie alla loro interconnessione funzionale, possono dunque essere impiegate insieme per facilitarne la diffusione e per creare un pacchetto tecnologico integrato di successo. Tra i fattori critici per l'adozione del car sharing elettrico, il contesto ambientale di sviluppo del servizio ha certamente un ruolo di primaria importanza.

La densità di popolazione incide soprattutto sul *free-floating* perché tale servizio è limitato alle grandi città, mentre i modelli che prevedono l'utilizzo di stazioni (*two-way* e *one-way*) possono essere estesi anche ad aree geograficamente più piccole. Inoltre risulta determinante analizzare quali sono i principali spostamenti degli utenti all'interno dell'area urbana affinché il car sharing possa soddisfare una domanda numerosa ed eterogenea. Per minimizzare l'utilizzo di vetture private, il servizio di condivisione delle auto elettriche deve essere integrato con un efficiente trasporto pubblico e un'infrastruttura ciclo-pedonale che incentivi lo sviluppo di una mobilità alternativa.

I veicoli elettrici introducono nuove sfide alla gestione del servizio a causa dell'autonomia limitata delle batterie, la quale condiziona la scelta di localizzazione delle stazioni di ricarica e comporta un continuo monitoraggio del livello di energia dei veicoli.

Le auto elettriche sono particolarmente adatte al car sharing perché, contrariamente ai veicoli a combustione, hanno una maggiore efficienza energetica nei percorsi urbani, dovuta all'elevata rigenerazione di carica che si ottiene in frenata. Secondo diversi studi, la massima quantità di energia che può essere reimmessa nelle batterie rallentando o percorrendo una discesa è compresa tra l'8% ed il 25%, a seconda di vari parametri tra i quali il tipo di percorso, la potenza ed il peso del veicolo. Lo stile di guida può giocare un ruolo importante nel

massimizzare l'economia delle batterie, così come l'attivazione di climatizzatore e riscaldamento incide notevolmente sul consumo.

La temperatura esterna è un altro fattore rilevante che influisce sull'autonomia di un'auto a propulsione elettrica: la resa chilometrica è ottimale intorno ai venti gradi centigradi, mentre diminuisce in modo considerevole, arrivando perfino a dimezzarsi, in condizioni climatiche più estreme (temperature superiori a trenta gradi o inferiori allo zero).

La standardizzazione di soluzioni tecnologiche a problemi che ostacolano la diffusione del servizio, come l'installazione di un'efficace infrastruttura di ricarica, rappresenta di certo un punto di svolta per favorire l'adozione di scenari alternativi a quelli tradizionali.

Tuttavia, il principale problema per l'avvio di un car sharing elettrico risiede nel fatto che mentre i vantaggi diretti della sua introduzione, come la drastica riduzione del livello di inquinamento, ricadono sulla collettività, gli svantaggi, come l'alto costo del servizio e le limitate prestazioni, restano a carico dell'utente. Finché non vengono intrapresi a livello governativo interventi massivi di incentivazione in grado di offrire vantaggi tangibili, risulta difficile invertire le tendenze attuali. Gli incentivi possono essere sia di tipo economico che non: oltre a tariffe agevolate, sarebbe opportuno garantire ai mezzi elettrici l'accesso libero e gratuito a zone a traffico limitato, corsie preferenziali e parcheggi a pagamento.

L'adozione dell'*e*-car sharing è influenzato anche da altri fattori legati al contesto di applicazione, tra i quali il prezzo del carburante, la connettività al servizio attraverso applicazioni su dispositivi mobili come gli *smartphone* e la positiva immagine sociale derivante dall'utilizzo di veicoli ecosostenibili.

Le caratteristiche individuali degli utenti sono altrettanto determinanti nel sostegno a nuove forme di mobilità alternativa, poiché la familiarità con la tecnologia implementata nel servizio (ad esempio le applicazioni web sviluppate dalle compagnie per la prenotazione, la localizzazione e l'utilizzo delle auto) e la conoscenza di esperienze positive da parte di altri membri, possono ridurre lo scetticismo iniziale verso tale innovazione.

La disponibilità di veicoli nel raggio di cinquecento metri dalla posizione del cliente è una condizione necessaria ma non sufficiente per l'efficacia del servizio, dato che le auto devono avere una carica residua che consenta agli utenti di percorrere i propri tragitti urbani. L'integrazione del car sharing elettrico con il trasporto pubblico o con altre forme di mobilità incrementa notevolmente l'efficacia del servizio. Secondo la maggior parte degli automobilisti, scegliere il modo più efficiente per andare da un punto a un altro equivale a determinare la

strada con il minor tempo di percorrenza, ma il futuro di cui si parla da anni è l'intermodalità, ovvero l'utilizzo di più mezzi di trasporto pubblici e privati.

Il progetto Ha:Mo (*Harmonious Mobility*) della città francese di Grenoble è un perfetto esempio di realizzazione di car sharing elettrico intermodale. Esso ha l'obiettivo di rendere il servizio complementare rispetto al sistema di trasporto pubblico locale: settanta auto disponibili in ventisette parcheggi, i quali sono anche stazioni di ricarica, possono essere prenotate utilizzando la stessa tessera con cui si può prendere l'autobus o il tram.

I parcheggi sono dislocati in modo funzionale alle stesse fermate dei mezzi pubblici e bisogna necessariamente riconsegnare il veicolo in un'altra stazione dedicata, collegandolo alla colonnina di ricarica.

Questo modello *one-way* ha dei limiti pratici rispetto a quello a flusso libero, ma è figlio dell'ambizione di incentivare l'uso del car sharing per percorrere solo il cosiddetto "ultimo chilometro" della mobilità individuale, dopo aver preso un mezzo pubblico. Infatti, attraverso l'applicazione ufficiale su *smartphone* o il sito internet del servizio, è possibile pianificare il proprio percorso intermodale, controllando in tempo reale gli orari dei trasporti e la disponibilità dei veicoli in car sharing, prenotabili fino a trenta minuti prima del noleggio.

1.4 *Le componenti strutturali del car sharing elettrico*

Affinché un generico servizio di condivisione di auto non elettriche risulti efficiente e competitivo, i seguenti parametri chiave devono essere analizzati e calibrati opportunamente:

- dimensione ottima della flotta e sua distribuzione nelle stazioni dedicate;
- localizzazione delle aree di parcheggio;
- politiche di tariffazione;
- modalità di accesso al servizio e tecnologie implementate;
- regolamento e condizioni di noleggio.

Se nella flotta si inseriscono delle vetture elettriche, è necessario considerare ulteriori criticità:

- numero e tipologia di stazioni di ricarica;
- localizzazione delle colonnine di ricarica;
- regolamento per la carica dei veicoli.

Un modello UML (*Unified Modeling Language*) può essere utilizzato per formalizzare gli aspetti sopra elencati e descrivere in modo sintetico l'organizzazione di un car sharing che comprende sia auto tradizionali che elettriche.

In particolare, un diagramma delle classi consente di rappresentare la struttura complessiva del servizio, i suoi elementi distintivi e le relazioni tra essi. L'elemento principale del diagramma è la classe, la quale rappresenta un insieme di entità chiamate oggetti ed il cui simbolo grafico è un rettangolo suddiviso in tre sezioni, rispettivamente dedicate al nome della classe (deve essere unico per distinguerle tra loro), agli attributi che caratterizzano gli oggetti e alle operazioni che ne descrivono il comportamento.

Le classi possono essere legate da associazioni che rappresentano le relazioni che sussistono fra gli oggetti delle classi ed ogni categoria di associazione (aggregazione, composizione, dipendenza, generalizzazione) viene rappresentata mediante una particolare freccia che connette le due classi coinvolte.

Tali associazioni possono inoltre essere corredate da informazioni relative alla molteplicità, cioè il numero di oggetti delle due classi associate che sono coinvolti nella relazione. Come si può notare nella figura sottostante, in un generico servizio di car sharing possiamo identificare le seguenti componenti strutturali:

− sistema di gestione: si occupa del monitoraggio e del coordinamento del servizio;

− sistema di gestione delle prenotazioni: gestisce le operazioni di prenotazione dei veicoli;

− sistema di gestione dei pagamenti: determina il costo del viaggio effettuato e controlla la corretta esecuzione della transazione;

− sistema di gestione delle attività di ricollocamento: monitora la distribuzione dei veicoli nei parcheggi e, se necessario, indica gli spostamenti necessari per bilanciare la flotta;

− sistema di gestione delle politiche di tariffazione: determina i costi del servizio ed eventuali agevolazioni;

− sistema di gestione delle emergenze: si occupa delle emergenze comunicate attraverso gli strumenti ICT installati a bordo dei veicoli;

− sistema di gestione delle registrazioni: gestisce la fase di iscrizione di nuovi membri;

− sistema di gestione delle manutenzioni: definisce i controlli ordinari e straordinari;

− compagnia di car sharing: società che gestisce il servizio;

− operatore: impiegato dell'organizzazione;

− utente: membro che, una volta iscritto, può prenotare un'auto;

− veicolo: macchina a disposizione nella flotta;

− veicolo tradizionale: sottoclasse di “veicolo” che rappresenta le auto con motore a combustione interna;

− veicolo elettrico: sottoclasse di “veicolo” che raggruppa le macchine a propulsione elettrica;

− area di parcheggio: stazione dedicata al servizio nella quale sono disponibili le auto;

− parcheggio tradizionale: sottoclasse di “parcheggio” che identifica il luogo in cui è possibile noleggiare e restituire le vetture;

− stazione di ricarica: sottoclasse di “parcheggio” che indica un’area di sosta provvista di un’infrastruttura di ricarica per veicoli elettrici.

Inoltre, nel diagramma possono essere individuate le seguenti associazioni tra classi:

− prenotazione: relazione tra le classi “utente”, “parcheggio” e “veicolo”;

− ricollocamento: relazione tra le classi “parcheggio”, “veicolo” e “operatore”;

− manutenzione: legame tra “veicolo” e “operatore”;

− emergenza: interconnessione tra “veicolo” e “operatore”;

− sostituzione/acquisto: acquisizione di nuova auto attraverso la relazione tra “veicolo” e “operatore;

− condivisione dell’auto: noleggio del veicolo condiviso tra diverse istanze della classe “utente”.

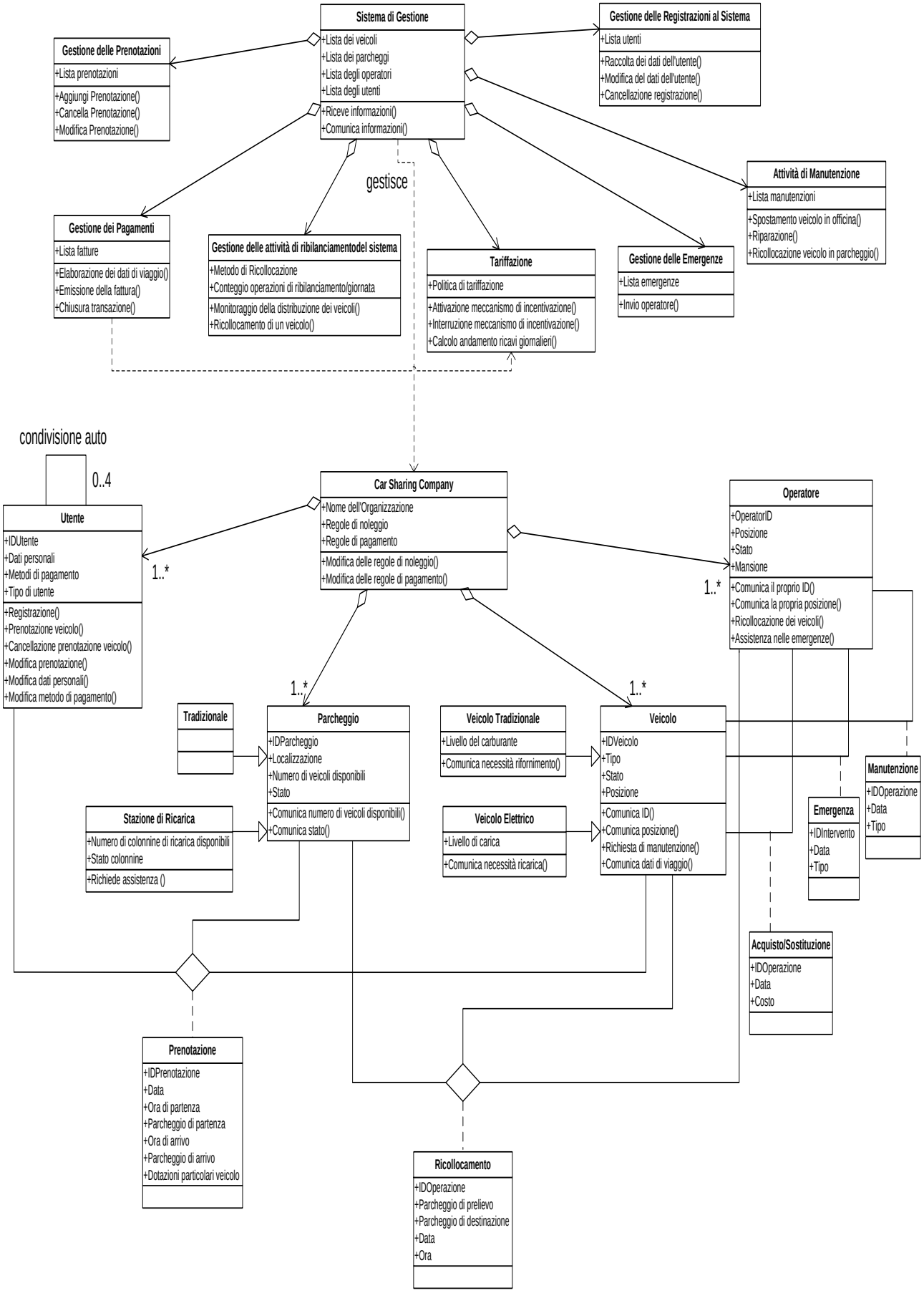


Figura 1: Diagramma delle classi per un generico servizio di car sharing

Lo stato del sistema può dunque essere rappresentato attraverso i valori degli attributi delle classi, mentre il comportamento dinamico del servizio di car sharing è descritto dal diagramma delle attività (anch'esso all'interno dello UML), il quale definisce l'insieme degli eventi che costituiscono un processo.

Tale diagramma è suddiviso in colonne, denominate *swimlane*, per evidenziare quale attore sia responsabile di una data azione.

Le attività, raffigurate con un rettangolo smussato contenente la loro descrizione, sono gli elementi principali dello schema ed il flusso, rappresentato tramite delle frecce orientate, indica la sequenza temporale con cui devono essere effettuate le attività. Queste ultime possono essere in parallelo e, in tal caso, il punto di divisione (*fork*) è raffigurato da frecce divergenti rispetto ad un segmento orizzontale; il punto di ricongiungimento (*join*) è reso sempre tramite un segmento orizzontale sul quale incidono le frecce.

Nel caso in cui le attività siano alternative, cioè svolte o meno rispetto ad una scelta, il punto di decisione è rappresentato da rombi dai quali partono i diversi flussi. L'introduzione di veicoli elettrici nel car sharing incrementa la complessità gestionale del servizio, in quanto i tempi di ricarica e di attesa di disponibilità di una colonnina costituiscono dei ritardi aggiuntivi che possono ridurre drasticamente il numero di domande soddisfatte (dunque anche i profitti della compagnia).

L'utente, il veicolo e il sistema di gestione sono gli attori del processo di noleggio, caratterizzato dalle seguenti fasi principali:

− richiesta di un'auto in un parcheggio da parte dell'utente;

− controllo della disponibilità di un veicolo nel parcheggio attraverso il sistema di gestione;

− noleggio e utilizzo dell'auto prenotata. Le attività di determinazione del tempo di viaggio e della stazione di arrivo fanno parte del processo decisionale dell'utente (eccetto il *two-way*);

− restituzione del veicolo;

− ricarica dell'auto. Ciascun veicolo, una volta riconsegnato nel parcheggio, attende che sia disponibile una colonnina e inizia il suo processo di ricarica;

− manutenzione delle auto che necessitano di controlli dopo il periodo di noleggio;

− pagamento del servizio, dopo il quale l'utente abbandona il sistema.

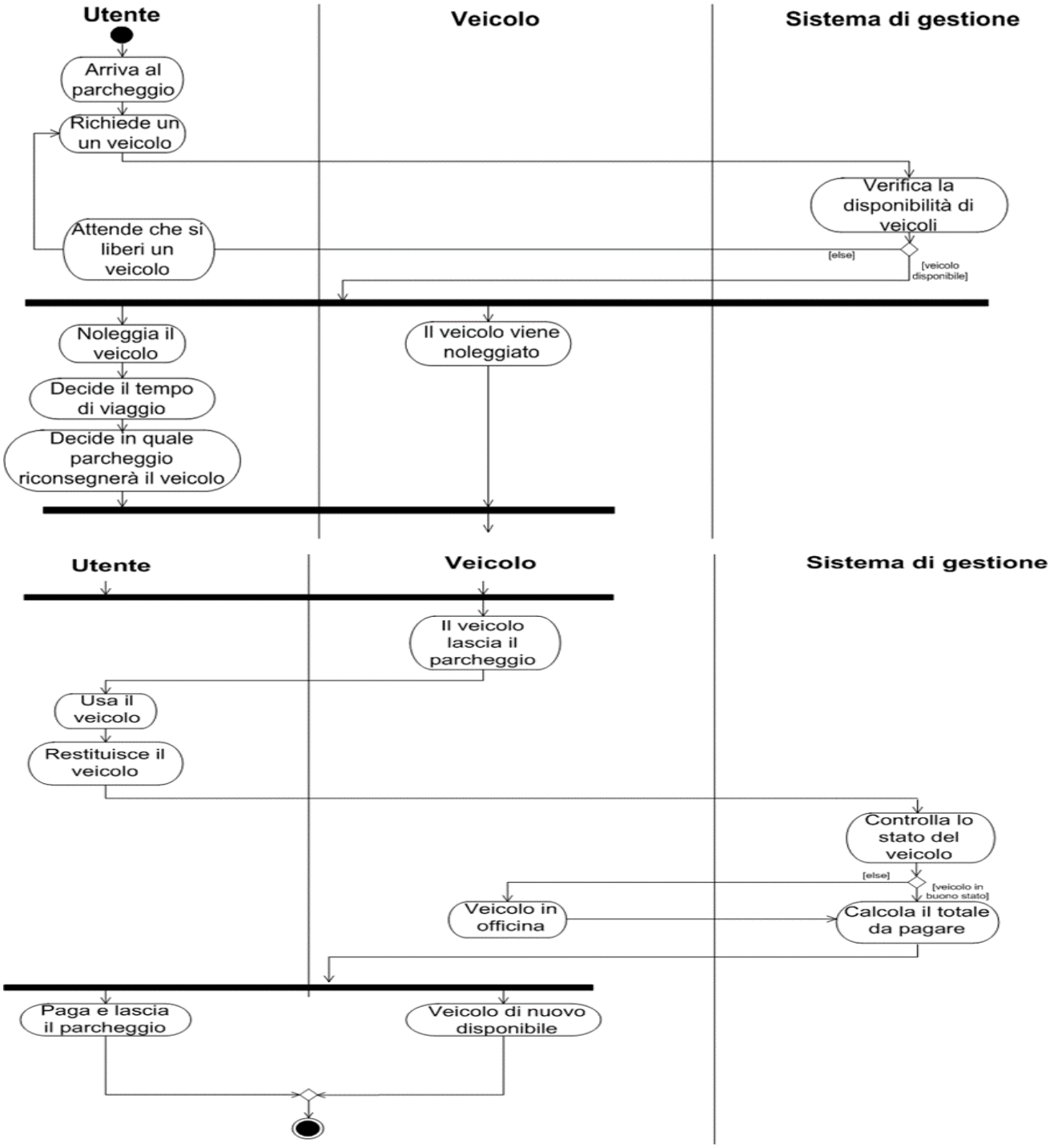


Figura 2: Diagramma delle attività per un generico servizio di car sharing

Capitolo 2
Il car sharing elettrico dell'Università di Bologna

Secondo l'ottavo rapporto elaborato lo scorso Dicembre da Euromobility con il Patrocinio del Ministero dell'Ambiente, Bologna è la città di riferimento in Italia per la mobilità sostenibile. Il capoluogo emiliano si è aggiudicato il titolo di città più "eco-mobile" non soltanto per la buona dotazione di parcheggi a pagamento e di scambio, per il basso indice di incidentalità sulle strade e la dotazione di piste ciclo-pedonali, ma anche per l'efficienza del trasporto pubblico e per il crescente servizio di car sharing.

La città di Bologna sta investendo molte risorse sullo sviluppo di una mobilità alternativa ed un'altra importante testimonianza di tale impegno è data dal progetto pionieristico denominato *City Logistic*, grazie al quale sarà possibile garantire lo spostamento di persone e merci sul territorio urbano con impatto zero.

L'obiettivo di questo programma, promosso dal CAAB (Centro Agroalimentare di Bologna) in collaborazione con l'amministrazione comunale e con un gruppo di aziende, è la creazione del primo circuito elettrico sostenibile d'Italia, il quale prevede il trasporto di merci (stimate a 800.000 chilogrammi all'anno) dal CAAB al centro storico cittadino tramite van elettrici, alimentati grazie all'energia prodotta dal più grande impianto fotovoltaico su tetto d'Europa, installato presso il Centro agroalimentare. Un progetto che consta di ben 43.750 pannelli solari e produce ogni anno 11.350.000 kWh che consentono il trasporto delle merci tramite il circuito elettrico urbano, facendo risparmiare ogni anno l'emissione in atmosfera di 10.000 chilogrammi di CO2.

L'importanza di servizi di condivisione di auto ecologiche si evince dall'analisi di alcuni dati sulla mobilità nella città di Bologna: circa 300.000 persone si muovono con mezzi pubblici o privati ed effettuano in media 2.000.000 di spostamenti complessivi giornalieri; 100.000 persone si muovono ogni giorno per lavoro, 40.000 per studio e 75.000 hanno destinazioni universitarie (la maggior parte è residente nella fascia suburbana o periferica); 410.000 sono gli spostamenti giornalieri interni al Comune con autovetture private e 147.000 i veicoli (auto, moto, mezzi pesanti, bus) circolanti nelle ore di punta.

Questi dati ci aiutano a prendere coscienza della congestione del traffico cittadino e delle conseguenti emissioni di agenti inquinanti dall'enorme impatto ambientale. In tale contesto si

colloca il contributo dell'Università di Bologna, il cui obiettivo è fornire un servizio di car sharing al proprio personale docente e tecnico-amministrativo, in grado di ridurre più del 90% delle precedenti emissioni di CO2. Le 74 vecchie auto di proprietà sono infatti state sostituite da 40 nuovi veicoli elettrici, ripartiti tra sei stazioni secondo le densità di domanda.

2.1 *I veicoli, le isole, le stazioni di ricarica e gli utenti*

Le auto sono state acquisite con la formula del noleggio tramite una convenzione Consip. Tutte sono equipaggiate di un sistema di georeferenziazione fornito da Telecom per la loro localizzazione e vengono riservate agli utenti attraverso un applicativo *web-based* che consente di verificarne la disponibilità. Per il nostro studio è necessario definire le caratteristiche tecniche delle autovetture, in quanto determinano la modalità di gestione della flotta. La capacità della batteria, l'autonomia, il consumo ed il tempo di ricarica sono dati critici che limitano gli spostamenti dei veicoli e dunque, per garantire un efficace servizio, vanno attentamente analizzati.

Cart, la società che si occupa dello sviluppo dell'applicativo dell'Università, sta attualmente implementando un sistema di controllo a distanza del livello di carica dei veicoli, al fine di rendere disponibili agli utenti soltanto le vetture con un'autonomia che soddisfi le loro richieste di viaggio. Di seguito sono riportate le specifiche dei tre modelli di veicoli elettrici utilizzati per il car sharing:

	Renault Kangoo	Citroen C0	BMW i3
Potenza (kW)	44	47	125
Coppia massima (Nm)	226	180	250
Capacità della batteria (kWh)	22	16	18.8
Autonomia (km)	170	150	160
Consumo (Wh/km)	129	125	129
Tempo di ricarica (h)	8	9	9
Velocità massima (km/h)	130	130	150

Tabella 1: Dati tecnici delle auto elettriche in car sharing dell'Università di Bologna

I veicoli elettrici possono circolare nelle zone a traffico limitato senza limiti temporali, parcheggiare gratuitamente nelle aree a pagamento ed accedere alle zone soggette a divieto di circolazione per provvedimenti anti inquinamento.

Figura 3: Le auto elettriche universitarie in car sharing

Le isole sono i luoghi fisici centralizzati nei quali sono aggregate le autovetture. Esse sono localizzate in corrispondenza dei principali insediamenti universitari di Bologna e presso i poli didattici di Forlì, Cesena e Rimini.

Il car sharing adottato è della tipologia *two-way*, dunque, al termine del viaggio gli utenti devono riconsegnare le auto nella medesima isola di appartenenza, salvo esigenze particolari da comunicare preventivamente agli operatori.

	Renault Kangoo finestrati	Renault Kangoo furgonati	Citroen C0	BMW I3	Totale
Belmeloro	1	2	6	1	**10**
Ingegneria	2	0	3	0	**5**
Filippo Re	2	0	2	0	**4**
Veterinaria (Ozzano)	1	1	0	1	**3**
CAAB	3	5	4	1	**13**
Poli di Forlì, Cesena e Rimini	1	0	2	2	**5**
Totale	**10**	**8**	**17**	**5**	**40**

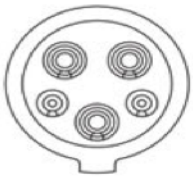
Tabella 2: Dislocazione delle auto nelle stazioni/isole

Ciascuna isola è dotata di punti di ricarica a 10 Ampère ed il fornitore è Edison Energia.

Sede	**Punti di ricarica**
Sede di Ingegneria, Via Risorgimento, zona portineria piano terra	6
Filippo Re 10, zona portineria piano terra	6
CAAB, primo piano	13
Sede di Ozzano, Via Tolare di Sopra 50	5
Belmeloro, Via Acri 10	11
Rimini, Via dei Teatini	2

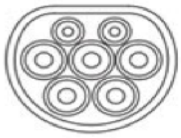
Confrontando i dati sopra riportati con la distribuzione delle auto nelle stazioni, possiamo notare che l'unica isola in difetto di punti di ricarica è quella dei poli universitari (due ricariche a fronte di cinque vetture), mentre le altre sedi, eccetto il CAAB, ne hanno uno o due in eccesso.

I veicoli inoltre necessitano di due diversi tipi di connettori:



Tipo 1 Renault Kangoo, Citroen C0

- 5 contatti: 3 di potenza, 2 di comunicazione (prossimità e controllo pilota)
- Monofase e trifase
- Max 32 A, 230 V



Tipo 2 BMW i3

- 7 contatti: 5 di potenza, 2 di comunicazione (prossimità e controllo pilota)
- Monofase e trifase
- Max 32 A, 230 V / 32 A, 400 V

Figura 4: Connettori utilizzati per la ricarica dei veicoli elettrici universitari

Nonostante questa differenza, per tutti i modelli di auto la massima potenza in ingresso è di 3,7 kW (16 Ampère e 230 Volt) e un'ora di ricarica consente di recuperare circa 26 chilometri di autonomia. Nel nostro caso con 2,3 kW di potenza (10 Ampère e 230 Volt) otteniamo 16 chilometri. In futuro l'Università vorrebbe dotarsi di colonnine di ricarica con supporto energetico da fonti rinnovabili, tuttavia al momento il progetto è lontano dalla sua realizzabilità.

Figura 5: Punti di ricarica nelle stazioni

Il servizio di car sharing è riservato al personale strutturato universitario, sia a tempo determinato che indeterminato, e a quello non strutturato con specifica autorizzazione del Dipartimento. Il personale universitario strutturato dell'Università di Bologna (ripartizione al 50% tra personale docente e tecnico-amministrativo) è composto da circa 6.000 unità, di cui 5.100 afferenti a Bologna. Da un'indagine in rete iniziata nel Giugno 2006, è possibile ricavare la percentuale di personale che effettua abitualmente spostamenti per motivi di lavoro nell'arco di una stessa giornata.

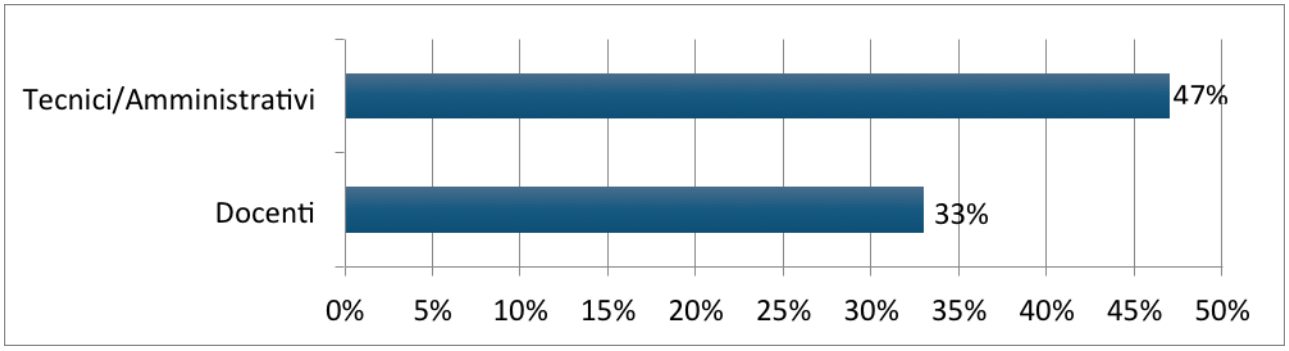


Figura 6: Percentuale di personale universitario che effettua giornalmente spostamenti per motivi di lavoro

I responsabili di plesso gestiscono le isole, verificano il regolare utilizzo dei mezzi e avvertono di eventuali disfunzionalità di parti meccaniche ed elettriche. Essi si occupano di manutenzione, gestione delle emergenze e dello spostamento dei veicoli tra le isole in caso di necessità.

2.2 *La gestione delle prenotazioni*

L'implementazione di un efficace sistema di prenotazione è alla base del funzionamento dell'intero servizio di car sharing elettrico. Il processo di gestione delle prenotazioni può essere formalizzato nei seguenti punti:

1. l'utente accede al software di prenotazione denominato "GoFlotte" tramite autenticazione con credenziali (login e password) di Ateneo;
2. ricerca di un veicolo disponibile sul calendario delle prenotazioni;
3. prenotazione di un veicolo specificando destinazione, ora di partenza e ora di ritorno;
4. il software comunica il codice di prenotazione a video e tramite email;

5. l’utente può recarsi al dispenser di pertinenza (l’isola del veicolo selezionato) e ritirare le chiavi dalla colonnina inserendo il codice;
6. al termine della prenotazione l’utente riconsegna le chiavi inserendo nuovamente il codice e selezionando l’opzione “riconsegna”;
7. l’utente è tenuto ad inserire il numero dei chilometri percorsi indicati nel veicolo al termine dell’utilizzo;
8. gli amministratori di sistema possono monitorare il processo dal pannello di amministrazione tramite report ed altri strumenti di gestione.

Le principali criticità monitorate nel corso della procedura sopra descritta sono:

- la coerenza dei chilometri inseriti alla riconsegna delle chiavi rispetto alle precedenti prenotazioni;
- verifiche e monitoraggi riguardo l’inserimento dei codici di prelievo/riconsegna delle chiavi (per esempio viene verificato il numero di tentativi falliti per evitare che qualcuno tenti di indovinare un codice);
- il ritardo nella riconsegna delle chiavi tramite invio di *email-alert* agli amministratori;

A fianco del software *open-source* di prenotazione, denominato “Booked” e opportunamente personalizzato, c’è un *bridge* sviluppato in ambiente Java che consente la comunicazione con i dispenser per il prelievo e la riconsegna delle chiavi. La tecnologia dei dispenser è invece sviluppata su piattaforma VisualBasic.

Figura 7: Dispenser per il prelievo e la restituzione delle chiavi

2.3 *Istanze per algoritmi di schedulazione di veicoli elettrici*

La generazione di istanze con i dati sulle prenotazioni degli utenti consente di testare l'algoritmo *greedy* sviluppato su numeri che rappresentano situazioni reali che il servizio dell'Università di Bologna si trova quotidianamente ad affrontare. I *file* di testo creati contengono i seguenti dati di prenotazione:

1) gli orari di partenza e arrivo dei veicoli;
2) le stazioni di competenza (con le lettere A,B,C,D,E,F sono indicate rispettivamente Belmeloro, CAAB, Filippo Re, Ingegneria, Veterinaria ad Ozzano dell'Emilia e i poli didattici di Forlì, Cesena e Rimini);
3) i chilometri percorsi per ciascun viaggio.

```
07 00   19 00   D   5
08 00   15 30   D   40
08 00   19 00   D   90
09 00   18 30   D   25
09 00   16 00   D   8
09 00   12 00   F   16
09 30   13 30   B   30
09 30   15 28   B   14
10 00   14 17   A   9
10 00   18 25   C   113
10 30   12 00   E   6
11 00   12 51   B   17
11 00   11 30   A   2
11 00   12 00   C   85
12 00   14 00   E   18
12 30   16 00   E   26
14 00   16 30   F   15
15 00   16 00   B   27
17 30   19 08   A   3
```

Figura 8: Istanza del giorno 01/10/2014

Le prenotazioni, tutte effettuate a partire dal 01/10/2014, sono suddivise per giorno di utilizzo del servizio e sono disposte secondo un ordine crescente dell'orario di inizio del viaggo. Soltanto per le isole di Veterinaria e dei poli non erano purtroppo disponibili dati, dunque ho inserito degli orari fittizi di utilizzo delle vetture, immaginando una frequenza di prenotazioni inferiore rispetto alle altre isole localizzate nella città di Bologna.

Le istanze costituiscono delle richieste reali, tuttavia ne ho create anche alcune più sfidanti che raggruppano prenotazioni effettuate in giorni differenti (per esempio ho aggiunto ai dati di un Lunedì alcune richieste del Lunedì della settimana successiva). Attualmente il sistema è in grado di gestire agevolmente l'assegnazione dei veicoli agli utenti, ma è importante osservare come cambierebbe la situazione all'aumentare della domanda. Le istanze sono state generate in riferimento al problema dell'assegnazione di veicoli con limitata autonomia di viaggio.

L'obiettivo di algoritmi per la risoluzione di tale problema è minimizzare il numero di auto necessarie (dunque il costo totale del servizio) a soddisfare la domanda giornaliera, considerando come vincoli gli orari di prenotazione dei viaggi e i chilometri di autonomia delle vetture. Come risultato ci si attende una schedulazione ammissibile per ciascun veicolo, ovvero una sequenza di viaggi che soddisfi i vincoli imposti e che minimizzi il numero di auto richieste. Un efficiente servizio di car sharing deve infatti movimentare il più possibile le proprie auto e ridurre i tempi di inattività nelle stazioni attraverso una corretta allocazione delle proprie risorse, volta a ridurre i costi di gestione operativa.

Capitolo 3

Un algoritmo greedy per il car sharing universitario

3.1 *Problemi di* set partitioning *e* interval partitioning

Il problema che vogliamo definire per il servizio di car sharing elettrico dell'Università di Bologna è di *set partitioning*. Il partizionamento di un insieme è un problema NP-difficile di fondamentale importanza nell'ottimizzazione combinatoria e le sue principali applicazioni includono la schedulazione di veicoli. Dati un insieme $I = [1, 2, ..., m]$ di m oggetti e un insieme $J = [1, 2, ..., n]$ di n indici, siano P_j, $j \in J$, n sottoinsiemi di I, a ciascuno dei quali è associato un costo c_j. L'insieme $\overline{J}$ è una partizione di I se:

- $\overline{J} \subseteq J$;
- $\bigcup_{j \in \overline{J}} P_j = I$;
- $P_i \cap P_k = \emptyset \qquad \forall i, k \in \overline{J},\ con\ i \neq k$.

Una partizione J^* di I è minima se $\sum_{j \in J^*} c_j \leq \sum_{j \in \overline{J}} c_j$, per ogni possibile partizione $\overline{J}$ di I. Il problema di *set partitioning* consiste dunque nel determinare una partizione minima J^* dell'insieme I. Esso può essere formulato come problema di ottimizzazione, introducendo una matrice binaria $A \in \Re^{n \times m}$ (*n* righe ed *m* colonne), detta matrice di appartenenza, così definita:

$$a_{ij} = \begin{cases} 1 & se\ l'elemento i\ appartiene al\ sottoinsieme\ P_j \\ 0 & altrimenti \end{cases}$$

Indichiamo con $x_j \in \{0, 1\}$, $j \in J$, le variabili decisionali aventi il seguente significato: $x_j = 1$ se l'indice j appartiene a J^*, $x_j = 0$ altrimenti. Una possibile formulazione del problema di *set partitioning* è dunque la seguente:

$$\begin{cases} z^* = \min \ \sum_{j \in J} c_j \, x_j \\ \qquad\quad \sum_{j \in J} a_{ij} \, x_j = 1 \quad i \in I \\ \qquad\quad x_j \in \{0,1\} \qquad j \in J \end{cases}$$

oppure, in forma compatta:

$$\begin{cases} z^* = \min \ c^T x \\ \qquad\quad Ax = e \\ \qquad\quad x_j \in \{0,1\} \quad j \in J. \end{cases}$$

con $c^T = [c_1, c_2, ..., c_n]$, $x^T = [x_1, x_2, ..., x_n]$, $e^T = [1, 1, ..., 1]$.

Nel nostro caso gli elementi dell'insieme da partizionare sono intervalli temporali, dunque nello specifico il problema è una variante dell'*Interval Partitioning Problem*, nel quale bisogna rispettare i vincoli energetici imposti dalla limitata autonomia dei veicoli elettrici.

L'IPP (*Interval Partitioning Problem*) classico è definito come segue: dato un insieme $\{(s(i), f(i)), 1 \le i \le n\}$ di *n* richieste da soddisfare, ciascuna delle quali è caratterizzata da un tempo di inizio $s(i)$ e uno di fine $f(i)$, vogliamo schedulare tutti gli intervalli utilizzando il minor numero possibile di risorse. Affinché una soluzione sia ammissibile, ciascun sottoinsieme deve contenere richieste compatibili tra loro, cioè non sono ammesse sovrapposizioni temporali. Se consideriamo gli intervalli in ordine crescente di $s(i)$, è possibile costruire per l'IPP classico un algoritmo *greedy* per minimo tempo di inizio (ESTF, *Earliest Start Time First*):

ORDINA intervalli per tempo di inizio: $s_1 \le s_2 \le ... \le s_n$
$d \leftarrow 0$ numero di risorse allocate
FOR i = 1 TO n
    IF intervallo *i* è compatibile con quelli assegnati alla risorsa *k*,
        assegna l'intervallo *i* a tale risorsa *k*
    ELSE
        Alloca una nuova risorsa $d + 1$, assegna l'intervallo i alla nuova risorsa $d + 1$.
        $d \leftarrow d + 1$
RETURN gli assegnamenti scelti.

La profondità di un insieme di intervalli è il massimo numero di intervalli che attraversano un dato istante comune t. Questo parametro risulta estremamente importante in quanto rappresenta un *lower bound* sull'ottimo, cioè, per ciascuna istanza di tale problema, il numero di risorse necessarie è pari almeno alla profondità dell'insieme di intervalli (risorse ≥ profondità).

Se infatti supponiamo la profondità d di un insieme di intervalli $I_1, ..., I_d$, ciascuno di essi deve essere schedulato con una risorsa differente dalle altre, pertanto ne vengono utilizzate almeno d. L'algoritmo *greedy Earliest Start Time First* non schedula mai due intervalli incompatibili nello stesso sottoinsieme ed è inoltre possibile dimostrare che risulta ottimale per un problema di partizionamento di intervalli classico:

- sia d il numero di sottoinsiemi che il *greedy* alloca;
- il sottoinsieme d è allocato la prima volta perché dobbiamo schedulare l'intervallo i che è incompatibile con tutti gli altri $d-1$ sottoinsiemi;
- quindi in ognuno di questi $d-1$ sottoinsiemi c'è un intervallo che finisce dopo $s(i)$;
- poiché questi $d-1$ intervalli sono già stati considerati dal *greedy* che schedula in ordine di tempo di inizio, vuol dire che tutti questi intervalli non cominciano più tardi di $s(i)$;
- includendo anche i, abbiamo d intervalli attivi subito dopo $s(i)$;
- dato che il numero di sottoinsiemi necessari è ≥ profondità, tutti gli algoritmi usano un numero di sottoinsiemi $\geq d$;

3.2 *Gli algoritmi* greedy

L'applicazione di metodi esatti a problemi di ottimizzazione combinatoria non è sempre possibile a causa della complessità intrinseca del problema (ad esempio, particolari problemi NP-Hard) e del tempo necessario per la sua risoluzione. In molti contesti è dunque opportuno sviluppare euristiche, ossia algoritmi che non garantiscono di ottenere l'ottimo della funzione obiettivo, ma in generale sono in grado di fornire una buona soluzione ammissibile.

Le euristiche costruttive costituiscono una delle principali classi di tecniche algoritmiche adottate perché risultano di facile implementazione e di notevole efficienza computazionale. La loro caratteristica essenziale è la progressività nella costruzione della soluzione: si parte da un insieme vuoto e si determinano in modo iterativo nuovi elementi da aggiungere fino ad arrivare ad una soluzione completa.

Se quest'ultima viene determinata secondo un criterio di ottimalità locale, si realizzano le cosiddette euristiche *greedy*, le quali adottano una logica di espansione basata sul miglioramento incrementale della funzione obiettivo. L'idea alla base di queste tecniche è di scegliere ripetutamente soluzioni localmente ottime per trovarne una globalmente ottima.

Lo schema concettuale di un algoritmo di questo tipo prevede dunque l'inizializzazione della soluzione e, per ogni scelta da effettuare, la selezione dell'elemento più conveniente, compatibilmente con i vincoli del problema. La grande diffusione degli algoritmi *greedy* è essenzialmente dovuta ai seguenti motivi:

- simulano quello che sarebbe il comportamento più intuitivo nella determinazione della soluzione;
- la loro implementazione risulta essere particolarmente semplice;
- il tempo di calcolo richiesto per determinare la soluzione è estremamente ridotto;
- forniscono blocchi da integrare in tutti gli algoritmi più sofisticati (ad esempio la soluzione iniziale per algoritmi di ricerca locale).

Gli algoritmi *greedy* sfruttano dei criteri di ordinamento degli elementi (*dispatching rule*), i quali prevedono di associare a ciascuna alternativa un valore di "appetibilità" che indichi la bontà della mossa, scegliendo così ad ogni iterazione quella che sembra essere la più promettente.

La logica di ordinamento viene calcolata all'inizio dell'esecuzione in base ai dati di input (pre-ordinamento), oppure può essere aggiornata dinamicamente tenendo conto delle scelte fatte in precedenza, con conseguente aumento del tempo di calcolo. Generalmente le euristiche *greedy* sono di tipo primale, ossia effettuano delle scelte che rispettino sempre i vincoli, tuttavia esistono anche delle versioni duali applicate a problemi per i quali può essere difficile determinare una soluzione ammissibile (ad ogni iterazione cercano di ridurre il grado di inammissibilità delle soluzioni).

Un algoritmo *greedy* può essere sinteticamente descritto attraverso il seguente schema generale:

```
INPUT I        (Istanza del problema)
    S ← S0                (Soluzione parziale iniziale per I)
    WHILE S può essere estesa DO
        Trova l'estensione locale S' di S più conveniente
```

S ← S’

ENDWHILE

OUTPUT S

3.3 *Definizione di un problema per il car sharing universitario*

Gestire una flotta totalmente elettrica aumenta la complessità della schedulazione dei veicoli a causa della limitata autonomia delle batterie, la quale vincola le auto a frequenti ricariche prima o dopo l’esecuzione di una richiesta di viaggio. Il servizio di car sharing dell’Università di Bologna può essere migliorato attraverso la formulazione e risoluzione di un problema di partizionamento degli intervalli temporali dei viaggi. Le richieste degli utenti sono suddivise per stazione prenotata ed ogni veicolo può essere utilizzato esclusivamente per soddisfare la domanda nell’isola di appartenenza.

Dato un insieme di *n* viaggi giornalieri, ciascuno dei quali caratterizzato da una lunghezza in chilometri e dai tempi di partenza e arrivo nella stazione selezionata, bisogna calcolare il costo minimo necessario a soddisfare tutte le richieste. Nel nostro caso consideriamo come costi i tempi di servizio dei veicoli nelle stazioni, con l’obiettivo di minimizzare il numero di auto assegnate ai viaggi.

L’intervallo temporale durante il quale è possibile accedere al car sharing universitario è compreso tra le ore sette e le venti, dunque ciascun veicolo è operativo per tredici ore al giorno. Nel servizio le vetture vengono sempre riportate nella stazione di prelievo (*two-way* car sharing) e alla consegna devono essere messe in carica. A parità di prenotazioni, il numero di veicoli elettrici utilizzato sarà dunque maggiore o uguale a quello delle normali vetture perché, oltre ai limiti temporali, dobbiamo considerare anche il vincolo energetico. La soluzione di tale problema è la sequenza di viaggi che ciascun veicolo deve compiere, minimizzando la dimensione della flotta necessaria a soddisfare la domanda giornaliera degli utenti.

Definiamo un insieme di *n* viaggi $T = \{t_1, t_2, ..., t_n\}$ e sei isole $D = \{\theta_1, \theta_2, ..., \theta_6\}$ nelle quali sono collocate altrettante stazioni di ricarica. Per ogni viaggio t_i è definita l’isola di prenotazione θ_j, i tempi di partenza st_i e arrivo et_i ed i chilometri da percorrere k_{t_i}. Assumiamo inoltre che l’autonomia massima di ciascun veicolo della flotta sia pari ad un valore costante ω pari a 150.

Come soluzione l'algoritmo calcola una schedulazione ammissibile di costo minimo, nella quale ciascun viaggio viene effettuato da un solo veicolo ed il numero di auto utilizzate in ogni isola è minimo. Al primo utilizzo giornaliero i veicoli hanno uno stato di carica uguale a 150, potendo essere ricaricati presso le stazioni per tutta la notte.

I viaggi richiesti dagli utenti sono definiti secondo un ordine crescente dell'orario di partenza, quindi $st_i \leq st_j$ per $1 \leq i \leq j \leq n$. Due viaggi $t_i, t_j \in T$ sono detti compatibili se $et_i \leq st_j$ (un veicolo può servirli entrambi senza violare i vincoli temporali) e se $a_{t_i} + f_{\theta_j} \geq k_{t_j}$, cioè l'autonomia residua della macchina al termine del viaggio t_i più quella recuperata durante la ricarica presso la stazione θ_j fino all'inizio del viaggio successivo, deve essere maggiore o uguale alla percorrenza richiesta da t_j.

Una schedulazione è dunque una successione di richieste giornaliere soddisfatte da un veicolo appartenente ad un deposito e la cui lunghezza deve essere almeno unitaria (deve essere effettuato almeno un viaggio). Una sequenza è ammissibile se il veicolo, attraverso l'eventuale utilizzo della stazione di ricarica, ha sufficiente autonomia per soddisfare tutte le domande assegnate.

Per ciascuna delle sei stazioni del car sharing universitario, indichiamo con Ω l'insieme di tutte le schedulazioni ammissibili dell'isola per un'istanza del nostro problema. Inoltre, per ogni schedulazione $p \in \Omega$, imponiamo la variabile v_p^i uguale ad 1 se p contiene il viaggio t_i e a 0 in caso contrario; c_p è il tempo di servizio giornaliero del veicolo associato a p nella stazione di appartenenza e x_p è una variabile binaria che vale 1 se includiamo p nella soluzione e 0 altrimenti. Attraverso queste variabili possiamo dunque definire per ogni stazione il seguente problema di programmazione lineare intera:

$$\text{Minimizzare} \qquad \sum_{p \in \Omega} c_p \, x_p \qquad (1)$$

$$\text{soggetto a} \qquad \sum_{p \in \Omega} v_p^i \, x_p = 1 \qquad i = 1, ..., n \qquad (2)$$

$$x_p \in \{0, 1\} \qquad \forall p \in \Omega \qquad (3)$$

L’obiettivo (1) è dunque minimizzare la somma dei costi operativi dei veicoli associati alle schedulazioni utilizzate nella soluzione ed il vincolo (2) assicura che ogni viaggio prenotato nella stazione venga effettuato da un solo veicolo.

3.4 *L’algoritmo*

L’algoritmo *greedy* sviluppato assegna le vetture elettriche in car sharing secondo una logica di minimizzazione delle risorse necessarie a soddisfare la domanda giornaliera. Il codice, scritto in linguaggio MATLAB, fornisce una schedulazione dei viaggi per ciascuna delle sei isole e l’ammissibilità della soluzione è garantita dall’esecuzione di tutte le attività e dal rispetto dei vincoli sia temporali che energetici.

La prima operazione effettuata è la lettura dei dati contenuti nelle istanze ed il loro conseguente inserimento in una matrice, la quale contiene dunque gli orari di inizio e fine delle attività, i nomi dei depositi di prenotazione e le percorrenze chilometriche richieste.

I viaggi sono elencati e stampati sul video esattamente come sono definiti nel calendario delle prenotazioni del servizio di car sharing universitario, cioè secondo un ordine crescente dell’ora di partenza. Come analizzato in precedenza nel partizionamento di un insieme, per modellare il problema dell’assegnamento consideriamo una matrice di appartenenza contenente elementi pari a 1 e a 0, a seconda che un viaggio venga o meno effettuato da un determinato veicolo.

Tale matrice, le cui righe e colonne rappresentano rispettivamente le auto allocate e le attività da svolgere, viene inizializzata attraverso una matrice identità, la cui dimensione è data dal numero dei viaggi; porre uguale ad 1 tutti gli elementi della diagonale principale e a 0 i restanti, equivale ad utilizzare un veicolo diverso per ogni richiesta degli utenti. Se ipotizziamo per ogni auto un costo operativo di tredici ore (il servizio è attivo dalle ore sette alle venti), il valore iniziale della funzione obiettivo è dunque pari al numero di viaggi o auto totali moltiplicato per tredici.

A questo punto avviamo il timer per calcolare il tempo impiegato dal greedy a risolvere il problema ed eseguiamo la schedulazione dei veicoli per ciascuna delle sei stazioni del servizio di car sharing. Infatti l’algoritmo innanzitutto suddivide le attività secondo le isole di prenotazione, in quanto queste ultime vanno gestite indipendentemente le une dalle altre, poi inizia a verificare la compatibilità dei viaggi per poterli effettuare con lo stesso veicolo.

Il primo controllo riguarda le sovrapposizioni degli intervalli temporali perché se due richieste si svolgono nello stesso momento, siamo certi di non poterle soddisfare con un unico mezzo. In caso contrario, invece, avviene un'ulteriore verifica sull'autonomia residua delle auto elettriche, la quale deve essere sufficiente a percorrere una nuova distanza.

Due parametri indispensabili per il calcolo energetico sono la capacità massima delle batterie e la loro velocità di ricarica: sulla base dei dati riportati in Tabella 1, per tutte e tre le tipologie di veicoli possiamo approssimarli rispettivamente a 150 chilometri e a 16 chilometri ogni ora.

Al primo utilizzo giornaliero, le auto hanno piena autonomia e, al termine del viaggio, occorre aggiornare lo stato di carica sottraendo la distanza percorsa; successivamente, poiché i veicoli possono recuperare energia presso le stazioni durante i periodi di inattività (il tempo che trascorre tra la fine di una richiesta e l'inizio della successiva), la carica della batteria viene calcolata come il valore minimo tra 150 ed i chilometri disponibili a seguito della sosta nel deposito.

Per ciascuna domanda, la variabile dell'autonomia dell'auto viene dunque aggiornata sulla base delle distanze percorse e delle ricariche effettuate in precedenza: se la carica è sufficiente per effettuare il viaggio, il veicolo può partire, altrimenti è necessario utilizzarne un altro.

L'algoritmo soddisfa il maggior numero possibile di richieste con un'unica risorsa e costruisce la soluzione finale incrementalmente, verificando ad ogni iterazione la possibilità di realizzare il viaggio j-esimo con il veicolo i-esimo e trasformando così l'iniziale matrice identità in una nuova di appartenenza.

Di seguito sono riportati il codice implementato ed i risultati ottenuti per una generica istanza del problema (il costo totale finale ed il numero di auto impiegate), incluso il grafico delle schedulazioni delle attività di ciascuna stazione. Quest'ultimo è costituito da due sezioni: quella sinistra rappresenta l'elenco dei viaggi da svolgere, mentre quella destra la soluzione finale, nella quale i sei diversi colori dei segmenti indicano le isole del servizio di car sharing elettrico universitario.

3.5 *Il codice*

File "solution_final.m"

```
% SCREEN INITIALIZATION AND MEMORY CLEARANCE
clear all; clc; close all
```

```
D = 6;  % Number of depots

% STRUCTURE FOR EACH TRIP
trips = [];
%----------read trips.txt file---------------
hFile = fopen('trips141014', 'r');
i=0;
while ~feof(hFile)
    i=i+1
    parsedStr = textscan(hFile,'%f %f %f %f %c %d');
    trips(i).ti = parsedStr{1}+1/60*parsedStr{2};
    trips(i).tf = parsedStr{3}+1/60*parsedStr{4};
    trips(i).depot = parsedStr{5};
    trips(i).distance = parsedStr{6};
end
fclose(hFile);

% INITIALIZATION OF THE ALLOCATION MATRIX
N = length(trips.depot); % number of activities
AllocMatrix = diag(ones(1,N));   % allocation matrix - rows -> vehicles,
% columns -> activities
AvailableTrips = 1:N;
% DISPLAY TRIPS
disp('LIST OF TRIPS')
for i = 1:N,
 print_trip(num2str(i),trips.depot(i),trips.ti(i),trips.tf(i),trips.distance(i))
end

% INITIAL NUMBER OF VEHICLES AND SCHEDULING
operating_time = 13; % daily service hours per vehicle
disp(['---> Initial Cost = ' num2str(operating_time*sum(sum(AllocMatrix'))) '...
Unitary Costs']); disp(' ');
disp([' Number of vehicles (initial): ' num2str(sum(sign(sum(AllocMatrix')))) ])
subplot(1,2,1); plot_org(AllocMatrix, trips, D); title('BEFORE')

disp('Start processing');
tic
for j = 'A':'F', % for all depots

    disp(['Depot ' j])
     trips_current_depot = [];

     for k = 1:length(AvailableTrips), % for all (still) available trips
         if(trips.depot(AvailableTrips(k)) == j)
            trips_current_depot = [trips_current_depot k];
         end
     end

     disp(trips_current_depot)  % trips for the current depot
     trips_left_in_current_depot = trips_current_depot;

     % for all trips in the depot
     for k = 1:length(trips_current_depot),
         rest_of_trips = [];
         last_time = trips.tf(trips_current_depot(k));

         for l = (k+1):length(trips_left_in_current_depot)
             % check if the other trips in the depot are compatible with
```

```
            % vehicle

            if((AllocMatrix(trips_current_depot(k),trips_current_depot(l))==0)...
             &&(AllocMatrix(trips_current_depot(l),trips_current_depot(l))~=0)..
             && is_compatible_trip(trips, trips_current_depot(k), ...
             trips_current_depot(l), AllocMatrix, last_time))
                  % exchange values
                  AllocMatrix(trips_current_depot(k),trips_current_depot(l))=1;
                  AllocMatrix(trips_current_depot(l),trips_current_depot(l))=0;
                  last_time=trips.tf(trips_current_depot(l));

            else
                % if not, create a list with the rest of the trips to
                % allocate to other vehicles
                rest_of_trips = [rest_of_trips trips_current_depot(l)];
            end
        end

        trips_left_current_depot = rest_of_trips;
    end

end
toc

disp(['---> Final Cost = 'num2str(operating_time*sum(sign(sum(AllocMatrix'))))...
' Unitary Costs']); disp(' ');
disp([' Number of vehicles (final): ' num2str(sum(sign(sum(AllocMatrix'))) ) ])
subplot(1,2,2); plot_org(AllocMatrix, trips, D); title('AFTER')
```

File "is_compatible_trip.m"

```
function flag = is_compatible_trip(trips, i, j, Mtrx, last_time)
% check if the trips don't overlap
if (trips.ti(j) >= last_time)
    % if they don't overlap, check the charge of the vehicle
    if (has_enough_charge(trips, i, j, Mtrx))
       flag = 1;
    else
       flag = 0;
    end
else
    flag = 0;
end
```

File "has_enough_charge.m"

```
function flag_charge = has_enough_charge(trips, i, j, Mtrx)

% constants
charge_max = 150;
charge_per_hour = 16;

charge = charge_max;
% check for all trips in the same depot how much charge is left when
```

```
% starting current trip
isFirst = 1;
prevStation = 0;
for aux = 1:(j-1),
    if( (isFirst == 1) && (Mtrx(i,aux) == 1) )
      charge = charge - trips.distance(aux); isFirst = 0;
      prevStation = aux;
    elseif(Mtrx(i,aux) == 1)
      charge = min(charge + charge_per_hour*(trips.ti(aux) - ...
      trips.tf(prevStation)), 150);
      charge = charge - trips.distance(aux);
      prevStation = aux;
    else
      % do nothing
    end
end

if(isFirst == 0)
   charge = min(charge+charge_per_hour*(trips.ti(j)-trips.tf(prevStation)),150);
end
% if the required charge is greater than the present charge, the vehicle cannot
% travel, otherwise yes

if(charge >= trips.distance(j))
    flag_charge = 1;
else
    flag_charge = 0;
end
```

File “print_trip.m”

```
function print_trip(id, trip_depot,trip_ti, trip_tf, trip_distance)
disp(['TRIP ' id ' - Depot: ' trip_depot ' Start: ' num2str(trip_ti)'h Stop:'...
      num2str(trip_tf) 'h Distance: ' num2str(trip_distance) 'km'])
```

File “plot_org.m”

```
function plot_org(Mtrx,  trips, n_depots)
colors = ['m' 'c' 'r' 'g' 'b' 'k']
clr_ = 1/n_depots;

for i=1:length(Mtrx(:,1)), %per vehicle

    for j=1:length(Mtrx(1,:)) %per task
        if(Mtrx(i,j) == 1)
            line([trips.ti(j) trips.tf(j)], ...
                [i i],'LineWidth',4,'Color',colors(trips.depot(j)-'A'+1), ...
                 'Marker', 'v' )
        end
    end

end

xlabel('Time')
ylabel('Task')
axis([6 22 0 (length(Mtrx(:,1))+1)])
```

Istanza con prenotazioni del 14/10/2014

07 30	19 00	D	5
08 00	13 00	A	43
08 00	15 00	F	52
08 30	13 30	D	9
08 30	12 30	D	22
08 30	16 30	D	46
08 30	16 20	B	69
08 30	17 06	B	6
09 00	11 32	B	18
09 00	17 30	D	39
09 00	13 00	A	7
09 00	12 30	E	20
09 30	14 16	B	44
09 30	14 00	F	37
10 00	13 00	E	16
10 30	13 41	B	17
13 00	16 00	D	4
14 00	15 46	B	11
14 00	16 30	C	9
14 00	17 30	D	3
14 00	18 00	E	31
14 30	17 00	F	15
15 00	18 00	A	10
16 00	18 00	F	7

3.6 *I risultati*

```
LIST OF TRIPS
TRIP 1 - Depot: D Start: 7.5h Stop: 19h Distance: 5km
TRIP 2 - Depot: A Start: 8h Stop: 13h Distance: 43km
```

```
TRIP 3 - Depot: F Start: 8h Stop: 15h Distance: 52km
TRIP 4 - Depot: D Start: 8.5h Stop: 13.5h Distance: 9km
TRIP 5 - Depot: D Start: 8.5h Stop: 12.5h Distance: 22km
TRIP 6 - Depot: D Start: 8.5h Stop: 16.5h Distance: 46km
TRIP 7 - Depot: B Start: 8.5h Stop: 16.3333h Distance: 69km
TRIP 8 - Depot: B Start: 8.5h Stop: 17.1h Distance: 6km
TRIP 9 - Depot: B Start: 9h Stop: 11.5333h Distance: 18km
TRIP 10 - Depot: D Start: 9h Stop: 17.5h Distance: 39km
TRIP 11 - Depot: A Start: 9h Stop: 13h Distance: 7km
TRIP 12 - Depot: E Start: 9h Stop: 12.5h Distance: 20km
TRIP 13 - Depot: B Start: 9.5h Stop: 14.2667h Distance: 44km
TRIP 14 - Depot: F Start: 9.5h Stop: 14h Distance: 37km
TRIP 15 - Depot: E Start: 10h Stop: 13h Distance: 16km
TRIP 16 - Depot: B Start: 10.5h Stop: 13.6833h Distance: 17km
TRIP 17 - Depot: D Start: 13h Stop: 16h Distance: 4km
TRIP 18 - Depot: B Start: 14h Stop: 15.7667h Distance: 11km
TRIP 19 - Depot: C Start: 14h Stop: 16.5h Distance: 9km
TRIP 20 - Depot: D Start: 14h Stop: 17.5h Distance: 3km
TRIP 21 - Depot: E Start: 14h Stop: 18h Distance: 31km
TRIP 22 - Depot: F Start: 14.5h Stop: 17h Distance: 15km
TRIP 23 - Depot: A Start: 15h Stop: 18h Distance: 10km
TRIP 24 - Depot: F Start: 16h Stop: 18h Distance: 7km


---> Initial Cost = 312 Unitary Costs
 Number of vehicles (initial): 24


Start processing
Depot A
     2    11    23

Depot B
     7     8     9    13    16    18

Depot C
    19

Depot D
     1     4     5     6    10    17    20
```

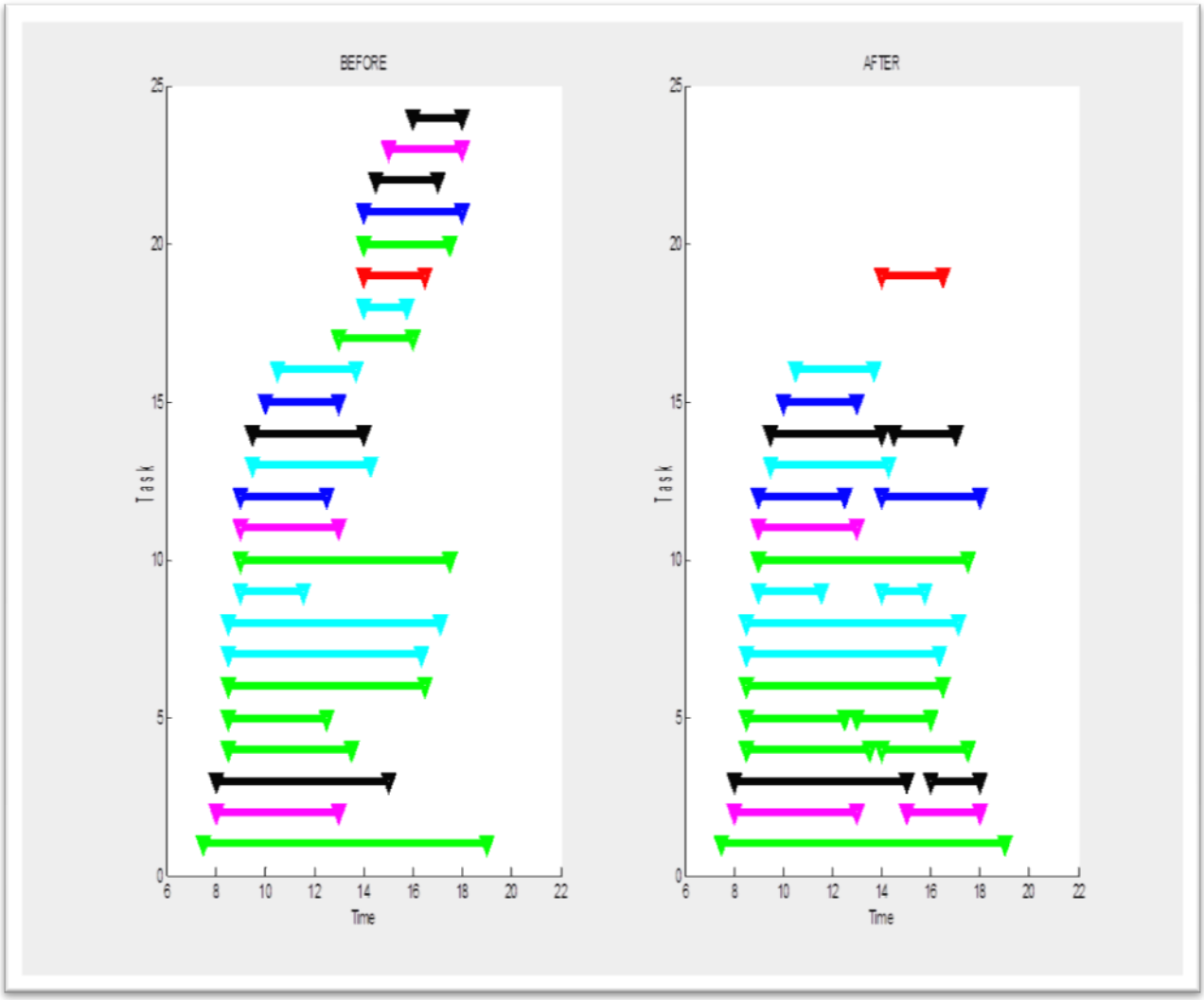
```
Depot E
    12    15    21

Depot F
     3    14    22    24


Elapsed time is 0.003202 seconds


---> Final Cost = 221 Unitary Costs
 Number of vehicles (final): 17
```

Grafico dei viaggi da effettuare e soluzione finale



Nel grafico sopra, i viaggi sono distinti per isola di appartenenza nel seguente modo:

Colore	**Isola**
Viola	Belmeloro
Celeste	CAAB
Rosso	Filippo Re
Verde	Ingegneria
Blu	Veterinaria
Nero	Poli

L'algoritmo è stato scritto in MATLAB, tutti i test sono stati eseguiti su un notebook con processore Intel Core i5-4210U 1.70 GHz, dotato di 4 GB di RAM e sistema operativo Microsoft Windows 8.1. Nella tabella sottostante sono riportati i risultati ottenuti per altre dieci istanze del problema.

Data delle prenotazioni	**Numero di viaggi**	**Numero di veicoli utilizzati**	**Costo totale**	**Tempo di calcolo (sec.)**
01/10/2014	19	14	182	0,002930
03/10/2014	21	16	208	0,002831
06/10/2014	20	15	195	0,002707
07/10/2014	19	14	182	0,004015
08/10/2014	21	17	221	0,003212
09/10/2014	26	15	195	0,003428
13/10/2014	20	14	182	0,002685
15/10/2014	25	18	234	0,003195
16/10/2014	26	20	260	0,003015
24/10/2014	24	21	273	0,003027

Tabella 3:Istanze e relativi risultati

Capitolo 4

Sviluppo di un modello di ottimizzazione

4.1 *Il metodo del* Branch and Bound

Descriviamo preliminarmente il metodo del *Branch and Bound*, poiché in questo capitolo vogliamo sviluppare e risolvere un modello di ottimizzazione attraverso un solutore che implementa tale tecnica. Un programma lineare intero è un problema di ottimizzazione vincolata in cui la funzione obiettivo è lineare e la regione ammissibile è costituita dall'insieme dei vettori interi (un vettore è detto intero se tutte le sue componenti sono intere) che soddisfano un dato sistema di equazioni e/o disequazioni lineari. Se scriviamo quest'ultimo nella forma compatta $Ax \sim b$, dove il simbolo "$\sim$" indica uno degli operatori "$\leq$", "$\geq$" o "$=$", $A \in \Re^{m \times n}$, $b \in \Re^m$ e $c \in \Re^n$, un problema di PLI è del tipo

$$\begin{aligned} z_I &= \min\ c^T x \\ s.t.\ \ & Ax \sim b \\ & x \in Z^n. \end{aligned} \qquad (1)$$

Tipicamente si tratta di problemi nei quali le variabili decisionali rappresentano quantità indivisibili, come il numero di automobili da utilizzare nel car sharing. Un programma lineare intero misto è la variante in cui la condizione di interezza è richiesta solo per un certo sottoinsieme delle variabili:

$$\begin{aligned} z_I &= \min\ c^T x \\ s.t.\ \ & Ax \sim b \\ & x_i \in Z,\ \ \forall i \in I, \end{aligned} \qquad (2)$$

per un qualche assegnato sottoinsieme $I \subseteq \{1, ..., n\}$. Le variabili x_i con $i \in I$ sono intere, le altre per cui $i \notin I$ continue.

Di fondamentale importanza in programmazione lineare intera (mista) sono le variabili binarie, cioè quelle che possono assumere solo valore 0 o 1: tali variabili vengono spesso usate

quando si ha la necessità di scegliere tra un numero finito di alternative diverse (decisioni del tipo sì/no). È sempre possibile imporre la condizione che una certa variabile x_i sia binaria scrivendo il vincolo nella forma $x_i \in \{0,1\}$. L'insieme

$$X = \{ x \in \Re^n \ : \ Ax \sim b, \ x \in Z^n \}$$

è la regione ammissibile del problema.

$$z_L = \min \ c^T x \qquad (3)$$
$$s.t. \quad Ax \sim b$$

è detto rilassamento lineare di (1), ottenuto eliminando dal problema di PLI il vincolo di interezza.

Si noti la seguente relazione:

$$z_L \leq z_I .$$

Infatti se x^I è la soluzione ottima di (1) e x^L è la soluzione ottima di (3), allora x^I soddisfa i vincoli di (3), dunque $z_L = c^T x^L \leq c^T x^I = z_I$. Il metodo più comune per risolvere problemi di programmazione lineare intera e sul quale si basa la maggior parte dei solver di PLI è il cosiddetto *Branch and Bound*. Consideriamo la seguente osservazione: data una partizione della regione ammissibile X in insiemi $X_1, ..., X_n$, sia $z_I^{(k)} = \min \{ c^T x : x \in X_k \}$.

Allora

$$z_I = \min_{k=1,..,n} z_I^{(k)} .$$

Il metodo di *Branch and Bound* procede partizionando X in sottoinsiemi più piccoli e risolvendo il problema $\min c^T x$ su ogni sottoinsieme. Questa procedura avviene dunque ricorsivamente attraverso la divisione delle regioni ammissibili dei sottoproblemi in sottoinsiemi. Se tale ricorsione venisse svolta completamente, alla fine enumereremmo tutte le possibili soluzioni intere del problema, ma questo presenta due problemi: prima di tutto, se il

problema avesse infinite soluzioni, l'enumerazione completa non sarebbe possibile, poi, anche se la regione ammissibile contenesse un numero finito di punti, esso potrebbe essere esponenzialmente grande, quindi enumerare richiederebbe un tempo eccessivo.

L'algoritmo di *Branch and Bound* cerca allora di esplorare solo aree "promettenti" della regione ammissibile, utilizzando dei limiti (*upper e lower bounds*) per escludere a priori alcuni sottoproblemi che sicuramente non contengono la soluzione ottima. Supponiamo di voler risolvere il problema (2) che denotiamo con (P_0).

L'algoritmo aggiorna la miglior soluzione ammissibile x^* per (P_0) (dunque $x_i^* \in Z$ per ogni $i \in I$) e l'upper bound UB sul valore ottimo z_I, dove $c^T x^* = UB$. Se denotiamo con ℓ l'indice massimo di un nodo (P_i) e con $\lfloor a \rfloor$ e $\lceil a \rceil$ rispettivamente l'arrotondamento per difetto e per eccesso di un numero a, il metodo genera un albero T in cui i nodi non eliminati rappresentano i sottoproblemi attivi. Di seguito è descritta la procedura di *Branch and Bound* per la ricerca di una soluzione ottima di un problema di PLI.

Inizializzazione: $T := \{(P_0)\}$, $\ell := 0$, $UB := \infty$, x^* non definito;

1. Se esiste un nodo attivo in T, scegline uno (P_k), altrimenti restituisci la soluzione ottima x^* e STOP;

2. Risolvi il rilassamento lineare di (P_k), determinando una soluzione ottima $x^{(k)}$ di valore $z_L^{(k)}$ oppure constatando che il problema è inammissibile. Ci sono tre possibilità:

   a) se il rilassamento lineare di (P_k) è inammissibile, elimina (P_k) da T;

   b) se $z_L^{(k)} \geq UB$, allora (P_k) non può avere soluzioni migliori di quella corrente x^*, dunque elimina (P_k) da T;

   c) se $x_i^{(k)} \in Z$ per ogni $i \in I$, allora $x^{(k)}$ è una soluzione ottima per (P_k) (ed ammissibile per (P_0)), dunque

      – Se $c^T x^{(k)} < UB$ (vero se non vale b)), poni $x^* := x^{(k)}$ e $UB := c^T x^{(k)}$;

      – Elimina (P_k) da T;

3. Se nessuno dei casi a), b), c) si verifica, allora scegli un indice $h \in I$ tale che $x_h^{(k)} \notin Z$, fai branching sulla variabile x_h, ponendo come figli di (P_k) in T i problemi

$$(P_{\ell+1}) := (P_\ell) \cap \left\{ x_h \le \lfloor x_h^{(k)} \rfloor \right\} \quad , \quad (P_{\ell+2}) := (P_\ell) \cap \left\{ x_h \ge \lceil x_h^{(k)} \rceil \right\}$$

$(P_{\ell+1})$ e $(P_{\ell+2})$ sono nodi attivi, (P_k) non è attivo. Poni $\ell := \ell + 2$ e torna ad 1.

Per rendere efficiente un metodo di *Branch and Bound*, è opportuno considerare i seguenti aspetti:

- *Soluzione del rilassamento ad ogni nodo*

Osserviamo che il rilassamento lineare da calcolare ad un nodo corrisponde al rilassamento lineare del nodo padre con l'aggiunta di un vincolo. Se abbiamo risolto il rilassamento del nodo padre con il metodo del simplesso, abbiamo a disposizione una soluzione ottima di base per il nodo padre.

Tramite una variante del metodo del simplesso chiamata "metodo del simplesso duale", si può ottenere in modo molto efficiente la soluzione ottima di un problema a cui è stato aggiunto un vincolo, a partire dalla soluzione ottima prima dell'aggiunta del vincolo stesso. Questo è uno dei fattori implementativi che accelera l'esplorazione dei vari nodi dell'albero di *Branch and Bound* per problemi di programmazione lineare intera (mista);

- *Scelta del nodo attivo*

Il passo 1 dell'algoritmo sopra esposto estrae un nodo dalla lista dei nodi attivi. Il numero di nodi che verranno aperti complessivamente dipende dalla gestione di questa lista e, in particolare, dai criteri di estrazione. Vi sono infatti due obiettivi contrastanti che concorrono alla scelta del nodo attivo:

– trovare rapidamente una (buona) soluzione intera ammissibile. Questo ha due vantaggi: una soluzione intera fornisce un *upper bound* al valore ottimo del problema che aumenta le possibilità di eliminare un nodo per *bound* e inoltre, qualora si decidesse di interrompere prematuramente il processo, abbiamo comunque determinato una soluzione ammissibile per il nostro problema iniziale;

– visitare il minor numero possibile di nodi.

Questi due obiettivi suggeriscono le seguenti strategie:

– *Depth-First-Search*

Questa strategia corrisponde ad una gestione LIFO (Last In First Out) della lista dei nodi attivi e, poiché fissa le variabili l'una dopo l'altra fino ad eliminare un nodo, consente di arrivare presto ad una soluzione intera.

Un altro vantaggio è che vengono mantenuti pochi nodi attivi e quindi mantenere la lista richiede poca memoria. Lo svantaggio è che si possono visitare nodi "poco promettenti", ove non vi siano soluzioni ottime o vicine all'ottimo;

– *Best-Node*

Per evitare di visitare nodi dove non vi siano buone soluzioni intere, una strategia è invece quella di scegliere un nodo attivo (P_k) con *lower bound* $z_L^{(k)}$ più piccolo possibile, ovvero $z_L^{(k)} = \min_t z_L^{(t)}$ ove il minimo è preso tra gli indici dei nodi attivi. In tal modo, siamo sicuri di non dividere un nodo il cui *lower bound* $z_L^{(k)}$ è maggiore del valore ottimo z_I di (P_0). Uno svantaggio di questa strategia è che richiede di mantenere molti nodi attivi, dunque molta memoria.

Una strategia ibrida è di applicare inizialmente *Depth-First-Search* per determinare in fretta una soluzione intera e poi passare ad una strategia di *Best-Node*. Inoltre, i solutori di programmazione lineare intera implementano solitamente euristiche per determinare in fretta una soluzione intera, prima di iniziare il procedimento di *Branch and Bound*.

- *Scelta della variabile di* branching

Anche in questo caso vi sono svariate strategie applicabili, tra le quali due molto comuni. La prima consiste nello scegliere la variabile di *branching* con la massima inammissibilità, cioè la cui parte frazionaria sia più vicina a 0,5. Posto $f_i = x_i^{(k)} - \lfloor x_i^{(k)} \rfloor$, scegliamo $h \in I$ tale che $h = \arg\min_{i \in I} \{\min\{f_i, 1 - f_i\}\}$. La seconda, invece, seleziona la variabile con la minima inammissibilità, cioè il cui valore sia più vicino a 0 o 1.

- *Valutazione di soluzioni ammissibili*

Per applicare efficacemente le regole di chiusura dei nodi è necessario disporre di soluzioni ammissibili di buona qualità, dunque bisogna stabilire come e quando calcolarle. Tra le varie possibilità possiamo:

– aspettare semplicemente che l'enumerazione generi un nodo ammissibile;

– implementare un algoritmo euristico che valuti una buona soluzione all'inizio, prima dell'esplorazione;

– sfruttare l'informazione raccolta durante l'esplorazione dell'albero, con frequenza da determinare in base al tipo di problema da risolvere, per costruire soluzioni ammissibili sempre migliori (ad esempio arrotondando la soluzione del rilassamento continuo in modo che risulti ammissibile).

In ogni caso, bisogna sempre valutare il compromesso tra la qualità della soluzione ammissibile corrente e lo sforzo computazionale per ottenerla.

- *Criteri di arresto*

Il metodo del *Branch and Bound* si arresta quando tutti i nodi sono dichiarati chiusi (la lista dei sottoproblemi attivi è vuota) ed in tal caso la soluzione ammissibile corrente corrisponde ad una soluzione ottima. Possono anche essere adottati criteri relativi a limiti computazionali, come ad esempio raggiunti limiti di tempo di calcolo o di memoria, ma non è garantito che l'eventuale soluzione ammissibile corrente sia ottima.

Notiamo che in ogni momento, durante la costruzione dell'albero, disponiamo non solo di un *upper bound* UB, ma anche di un *lower bound* LB, dato dal massimo di tutti i valori $z_L^{(k)}$ relativi a nodi attivi; tale valore è una stima ottimistica sul valore ottimo intero z_I, nel senso che necessariamente $z_I \geq LB$. Se decidiamo di interrompere l'algoritmo anticipatamente, la differenza tra UB e LB fornisce una stima della qualità della soluzione ammissibile a disposizione. Sfruttando proprio il fatto che si ha sempre a disposizione sia un *upper bound* che un *lower bound*, si potrebbe adottare un criterio di arresto basato sulla differenza tra questi valori: ci si ferma quando si ritiene "sufficiente" la qualità della soluzione disponibile, cioè la differenza percentuale tra UB e LB è sotto una soglia predefinita.

4.2 *Risoluzione di un problema di PLI con l'*Optimization Toolbox *di MATLAB*

L'*Optimization Toolbox* di MATLAB offre funzioni per trovare soluzioni che minimizzino o massimizzino la funzione obiettivo e allo stesso tempo soddisfino i vincoli. Il *toolbox* include solutori per la programmazione lineare, la programmazione lineare intera, la programmazione quadratica, l'ottimizzazione non lineare e i minimi quadrati non lineari.

Questi solutori possono essere utilizzati per trovare soluzioni ottimali a problemi continui e discreti, per eseguire l'analisi di *trade-off* e incorporare metodi di ottimizzazione in algoritmi e applicazioni. Le funzioni dell'*Optimization Toolbox* sono accessibili in modo programmatico o dall'applicazione di ottimizzazione, la quale consente di:

– selezionare un solutore e definire un problema di ottimizzazione;

– impostare e ispezionare le opzioni di ottimizzazione con i rispettivi valori predefiniti per il solutore selezionato;

– eseguire i problemi e visualizzare i risultati intermedi e finali;

– importare ed esportare le definizioni dei problemi, le opzioni dell'algoritmo e i risultati tra il *workspace* MATLAB e l'applicazione di ottimizzazione;

– generare automaticamente il codice MATLAB per registrare il lavoro svolto e automatizzare le operazioni.

Nell'*Optimization Toolbox* di MATLAB la funzione *intlinprog* consente di risolvere problemi di programmazione lineare intera mista della forma:

$$\min \quad c^T x$$

$$s.t \begin{cases} A x \le b \\ D x = e \\ l \le x \le u \\ x(intcon) \in Z \end{cases}$$

dove c è il vettore dei coefficienti della funzione obiettivo da minimizzare, x il vettore delle incognite, *intcon* il vettore che indica le componenti della variabile decisionale x che assumono valori interi, b ed e i vettori dei vincoli di disuguaglianza ed uguaglianza, l ed u i vettori dei limiti inferiori e superiori delle variabili, A e D le matrici dei coefficienti dei vincoli di disuguaglianza ed uguaglianza (tante righe quanti sono i vincoli e tante colonne quante sono le variabili). La sintassi del solutore è la seguente:

$$x = \text{int}linprog\ (c, \text{int}con, A, b, D, e, l, u, options)$$

L'algoritmo utilizzato in MATLAB per la programmazione lineare intera (mista) può risolvere i problemi in una delle seguenti fasi, oltre la quale non viene eseguito:

1. Riduce la dimensione del problema attraverso la pre-elaborazione del programma lineare.

2. Risolve un iniziale problema rilassato (non intero) con la programmazione lineare.

3. Esegue la pre-elaborazione del programma intero (misto) per restringere il rilassamento del problema intero (misto).

4. Applica piani di taglio per restringere ulteriormente il rilassamento del problema intero (misto).

5. Cerca soluzioni ammissibili intere attraverso un'euristica.

6. Utilizza un algoritmo di *branch and bound* per trovare la soluzione ottima.

1. Pre-elaborazione del programma lineare

Nel modello di programmazione lineare intera (mista), le matrici A, D ed i corrispondenti vettori b, e definiscono un insieme di disuguaglianze ed uguaglianze lineari che vincolano la soluzione x.

Solitamente è possibile ridurre il numero sia delle variabili (il numero delle componenti del vettore x) che dei vincoli lineari e l'esecuzione di tali riduzioni, nonostante richieda del tempo al solutore, consente di diminuire quello totale necessario a giungere alla soluzione.

Tali algoritmi sono inoltre in grado di rendere la soluzione numericamente più stabile e talvolta possono individuare l'inammissibilità del problema. Le fasi di pre-elaborazione hanno lo scopo di eliminare le variabili ed i vincoli ridondanti, migliorare lo *scaling* del modello e la sparsità della matrice dei vincoli, rafforzare i limiti sulle variabili e rilevare l'inammissibilità dei problemi primale e duale.

2. Programmazione lineare

Il rilassamento iniziale è un problema di programmazione lineare con la stessa formulazione di quello di programmazione intera mista, ma senza i vincoli di interezza sulle variabili. Indichiamo con x_{LP} la soluzione del rilassamento e con x quella del problema originale con le variabili intere.

Chiaramente $c^T x_{LP} \leq c^T x$ perché x_{LP} minimizza la stessa funzione obiettivo con minori restrizioni. Questo rilassamento iniziale e tutti i successivi generati dall'algoritmo di *branch and bound* sono risolti attraverso le tecniche di soluzione della programmazione lineare, come l'algoritmo del simplesso.

3. Pre-elaborazione del programma intero (misto)

Durante questa fase, *intlinprog* analizza i vincoli $Ax \leq b$ e di interezza per determinare se:

– il problema è inammissibile;

– alcuni limiti possono essere più stringenti;

– alcune disuguaglianze sono ridondanti, dunque possono essere ignorate o rimosse;

– alcune disuguaglianze possono essere rafforzate;

– alcune variabili intere possono essere fissate.

L'obiettivo principale della pre-elaborazione del programma intero misto è semplificare i calcoli al *branch and bound* attraverso una veloce esplorazione dell'area ammissibile e l'eventuale eliminazione di sottoproblemi futili (nessuna soluzione migliore dell'ottimo corrente può essere contenuta nell'insieme ammissibile del sottoproblema) che altrimenti verrebbero successivamente analizzati.

4. Piani di taglio

I tagli sono vincoli lineari di disuguaglianza addizionali che la funzione *intlinprog* aggiunge al problema. Tali vincoli cercano di restringere l'area ammissibile dei rilassamenti in modo tale da rendere le loro soluzioni più vicine ai valori interi.

5. Euristiche per la ricerca di soluzioni ammissibili

Per individuare un limite superiore al valore della funzione obiettivo, l'algoritmo di *branch and bound* deve trovare punti ammissibili. Durante l'esecuzione di quest'ultimo, la soluzione di un problema rilassato può essere intera e può fornire un migliore *upper bound* al problema originale di MILP.

Ci sono tecniche che, sebbene possano fallire nella ricerca, consentono di calcolare più velocemente soluzioni ammissibili prima e/o durante il *branch and bound*: le euristiche. E' possibile definire le seguenti euristiche:

- 'rins' ⇒ *intlinprog* cerca una nuova e migliore soluzione nel *neighborhood* della miglior soluzione corrente ammissibile intera (se disponibile);
- 'rss' ⇒ viene applicata una procedura ibrida che combina la ricerca di 'rins' ed il *local branching* per trovare soluzioni ammissibili intere;

- ‘round’ ⇒ *intlinprog* prende la soluzione del rilassamento lineare e ne arrotonda le componenti a valori interi in modo tale da mantenere l’ammissibilità;
- ‘none’ ⇒ non si cerca alcuna soluzione attraverso le euristiche, ma vengono presi in considerazione soltanto i punti ammissibili trovati con il *branch and bound*.

6. *Branch and Bound*

Applicazione del metodo del *Branch and Bound*.

4.3. *Definizione del modello di ottimizzazione*

L’algoritmo sviluppato nel capitolo precedente restituisce il numero minimo di veicoli necessari in ogni stazione per soddisfare la domanda giornaliera degli utenti.

A partire da tali risultati, vogliamo ora valutare i costi associati all’utilizzo di un minor numero di auto rispetto a quante ne sono richieste e, a tale scopo, definiamo le seguenti variabili: dati n giorni per $t = 1,..., n$ ed m stazioni per $d = 1,..., m$, $\forall t, d$, indichiamo con v_{td} il numero minimo di veicoli restituito dall’euristica *greedy*, con k la dimensione della flotta da valutare e con $cost_{tdk}$ il costo dei viaggi non effettuati del giorno t nella stazione d se utilizziamo k veicoli. Per $0 \le k < v_{td}$ si ha $cost_{tdk} > 0$ perché non siamo in grado di servire tutti i clienti, mentre per $k \ge v_{td}$ il costo è nullo ($cost_{tdk} = 0$).

Le auto vengono impiegate diversamente in termini di numero di utenti serviti, durata dei viaggi effettuati e chilometri percorsi, pertanto, sulla base della schedulazione ottenuta dall’algoritmo, associamo ad ogni veicolo un valore di “appetibilità”. In tal modo, a partire da $k = v_{td}$ fino ad arrivare a $k = 0$, è possibile costruire una matrice tridimensionale dei costi delle richieste non soddisfatte, escludendo di volta in volta l’auto di minor costo ($\forall t, d$, $cost_{td0}$ è pari al costo di tutti i viaggi del giorno t nella stazione d).

Dunque, secondo tale logica, i primi veicoli ai quali rinunciare sono quelli che effettuano pochi viaggi o che compiono tragitti di breve durata e lunghezza, poiché è ragionevole considerarli meno strategici all’interno della flotta a disposizione. Le istanze analizzate in questo nuovo problema non costituiscono delle richieste reali come quelle rappresentate in Tabella 3, ma sono più sfidanti, cioè contengono un maggior numero di viaggi (tra i settanta e i

novanta circa ciascuna), in quanto siamo interessati allo studio dell'assegnazione delle auto all'aumentare della domanda giornaliera.

La dimensione k della matrice è pari a 40, cioè alle auto totali disponibili, mentre il numero di stazioni è uguale a sei ($m = 6$). Una volta ricavata la matrice dei costi degli utenti non serviti, la utilizziamo come parametro in un nuovo problema di Programmazione Lineare Intera (PLI). Oltre a costituire una descrizione formale della situazione, il modello fornisce la base per l'applicazione di algoritmi in grado di determinare una soluzione ottima, ricavata calcolando i valori delle variabili che soddisfano tutti i vincoli e minimizzano la funzione obiettivo.

Il car sharing dell'Università di Bologna comprende quaranta veicoli elettrici ed i potenziali utenti del servizio sono circa seimila, dunque è necessario modellare un problema di distribuzione giornaliera della flotta nelle isole a fronte di un elevato numero di prenotazioni.

Oltre ai costi di mancato servizio dei clienti, un altro aspetto da minimizzare sono le spese di ricollocamento delle auto, poiché movimentarle ogni giorno da una stazione all'altra richiede la disponibilità di un gruppo di operatori da dedicare al servizio. Il periodo temporale lungo il quale valutiamo i costi totali è n, in ognuno dei quali si deve stabilire il numero ottimale di veicoli da allocare nei sei depositi.

Diamo di seguito la formulazione del problema, individuando gli elementi che compongono un modello di programmazione matematica. Definiamo innanzitutto gli insiemi delle entità coinvolte:

- T: giorni delle prenotazioni;
- D: stazioni del servizio;
- K: veicoli utilizzati.

I parametri sono:

- $cost_{t\,d\,k}$: costo degli utenti non serviti del giorno $t \in T$, nella stazione $d \in D$ se utilizziamo k veicoli;
- $kmax$: numero di veicoli disponibili;
- k: numero di veicoli utilizzati;

Le variabili decisionali:

- $x_{t\,d\,k}$: variabile binaria che vale 1 se nel giorno t per il deposito d utilizzo k veicoli, 0 altrimenti;

- $z_{t\,d}$: numero di auto da spostare il giorno $t \in T$ nella stazione $d \in D$.

Il modello è il seguente:

$$\min \quad \sum_{t=1}^{n} \sum_{d=1}^{m} \sum_{k=0}^{k\max} \text{cost}_{t\,d\,k}\; x_{t\,d\,k} \;+\; \sum_{t=1}^{n} \sum_{d=1}^{m} z_{t\,d}$$

$$s.t. \quad \sum_{d=1}^{m} \sum_{k=0}^{k\max} k\; x_{t\,d\,k} \;\le\; k\max \qquad \forall\; t \in T$$

$$\sum_{k=0}^{k\max} x_{t\,d\,k} \;=\; 1 \qquad \forall\; t \in T,\, d \in D$$

$$\sum_{k=0}^{k\max} k\; x_{t\,d\,k} \;-\; \sum_{k=0}^{k\max} k\; x_{t-1\,d\,k} \;\le\; z_{t\,d} \qquad \forall\; t \in T,\, d \in D$$

$$x_{t\,d\,k} \;\in\; \{0,1\} \qquad \forall\; t \in T,\, d \in D,\, k \in K$$

$$z_{t\,d} \;\in\; Z^{+} \qquad \forall\; t \in T,\, d \in D$$

La funzione obiettivo minimizza i costi di mancato servizio degli utenti e quelli di spostamento dei veicoli tra le isole; in questo nuovo problema vogliamo infatti valutare le due voci di spesa, al fine di migliorare l'efficienza del processo di assegnazione delle auto.

Il primo vincolo garantisce che il numero totale di vetture utilizzate ogni giorno nelle sei stazioni non superi la dimensione della flotta, mentre il secondo assicura che in ciascun deposito venga allocata quotidianamente una sola quantità di veicoli.

Il terzo vincolo definisce il numero di auto da movimentare in una stazione come la differenza tra quelle utilizzate in un giorno e nel precedente; $z_{t\,d}$ appartiene all'insieme dei numeri interi positivi ed assume il valore zero nel caso in cui il risultato della differenza sia negativo.

Lo scenario che modelliamo prevede dunque una gestione flessibile della flotta, nella quale i veicoli richiesti e non disponibili nelle isole vengono forniti dagli operatori che li spostano laddove necessario (possiamo ipotizzare che il trasferimento delle auto avvenga all'orario di chiusura del servizio di car sharing).

Per implementare il modello, occorre prima di tutto costruire la matrice dei costi degli utenti non serviti, costituita dagli elementi $cost_{t\,d\,k}$. Questi ultimi possono essere definiti associando un costo a ciascun veicolo utilizzato, secondo uno dei seguenti tre criteri: numero di richieste soddisfatte, durata totale dei viaggi effettuati e somma delle distanze percorse.

Per ogni deposito definiamo infatti un vettore riga avente tante colonne quante sono le auto impiegate nella stazione, dopodiché vi associamo i relativi costi.

A questo punto disponiamo gli elementi del vettore in ordine crescente ed eseguiamo la somma cumulata di essi, in modo tale da ottenere un nuovo vettore avente una somma progressiva dei valori.

Creiamo una matrice di zeri avente un numero di righe pari al numero di stazioni (6) ed un numero di colonne pari al numero di veicoli che è possibile utilizzare (40), poi la riempiamo nel seguente modo: se i veicoli del deposito *i*-esimo sono minori o uguali a 40, inseriamo dalla prima all'ultima colonna della riga *i*-esima gli elementi del vettore della somma cumulata in ordine invertito; altrimenti, se le auto sono più di 40, disponiamo i valori del vettore sempre in modo invertito, cioè dal più grande al più piccolo, inserendo gli elementi fino a riempire le colonne disponibili.

La matrice così ottenuta è coerente con la disposizione desiderata dei costi di mancato servizio, infatti essi sono maggiori al diminuire del numero di veicoli utilizzati.

Questa procedura viene eseguita per ogni giorno di prenotazione, dunque $\forall t \in T$ definiamo una nuova matrice dei costi 6×40, in modo tale da ottenere una matrice finale tridimensionale contenente i valori $cost_{t\,d\,k}$ associati all'utilizzo di k veicoli nel deposito d nel giorno t.

Come abbiamo descritto nel paragrafo riguardante la sintassi del solutore "intlinprog" di MATLAB, per implementare il modello di ottimizzazione sopra formulato è necessario determinare i dati in ingresso. Tra le diverse strutture di *input* disponibili, quella che ci consente di risolvere il problema è la seguente:

$$x = \text{int}linprog\ (c, \text{int}con, A, b, D, e, l, u, options)\,.$$

Il vettore c dei coefficienti della funzione obiettivo da minimizzare è costituito dall'insieme dei costi di mancato servizio $cost_{t\,d\,k}$ e dai coefficienti 1 che moltiplicano le variabili $z_{t\,d}$.

Dato che il problema formulato è di PLI, in *intcon* inseriamo tutte la variabili decisionali, in modo tale che assumano valori interi. La matrice A dei coefficienti dei vincoli di disuguaglianza contiene tutti i possibili valori di k, affinché si garantisca che il numero totale di veicoli utilizzati ogni giorno nelle sei stazioni non sia superiore alla dimensione della flotta, ossia al vettore b.

Nelle matrici A, D (costituita da coefficienti 0 e 1 che definiscono il vincolo di uguaglianza, secondo il quale ogni giorno in ogni stazione può essere definito un solo numero k di veicoli) e nel vettore b dobbiamo inoltre modellare il vincolo sul numero di auto da spostare giornalmente in ogni stazione ($z_{t\,d}$).

Il vettore colonna e ha un numero di righe pari al prodotto tra i giorni di prenotazione e i depositi del servizio. Per rendere le variabili $x_{t\,d\,k}$ binarie dobbiamo impostare un *lower bound* l pari a 0 ed un *upper bound* u pari a 1, mentre le variabili $z_{t\,d}$ hanno *lower bound* l uguale a 0 ed *upper bound* u pari a $k\max$. Successivamente impostiamo con il comando *options* la strategia di branching che seleziona la variabile con la massima inammissibilità e definiamo come soluzione iniziale $x0$ la matrice corrispondente all'allocazione giornaliera di 0 veicoli in tutti i depositi, con conseguente costo massimo.

4.4 *Il codice*

Di seguito è riportato il codice MATLAB per lo sviluppo del modello di ottimizzazione:

File "solution_final.m"

```
% SCREEN INITIALIZATION AND MEMORY CLEARANCE
clear; clc; close all
criteria = 2;
weights = [0 1];

verbose = 0; % print events
verbose_debug = 0;
```

```
D = 6;  % Number of depots
N_days = 3;  % Number of days
Kmax = 40; % Number of vehicles

% STRUCTURE FOR EACH TRIP
Cost = zeros( D, Kmax, 1);
realday = 0;
disp('Start  processing');
tic

for day = 1:N_days,

      trips = [];
      %----------read trips.txt file---------------
      Filename = sprintf('trips%02d1014.txt', day);
      hFile = fopen(filename, 'r');
      if hFile == -1, continue; end
      realday = realday+1;
      disp(filename);
      i=0;
      while ~feof(hFile)
            i = i+1
            parsedStr = textscan(hFile,'%f %f %f %f %c %d');
            trips(i).ti = parsedStr{1}+1/60*parsedStr{2};
            trips(i).tf = parsedStr{3}+1/60*parsedStr{4};
            trips(i).depot = parsedStr{5};
            trips(i).distance = parsedStr{6};
      end
      fclose(hFile);

      % INITIALIZATION OF THE ALLOCATION MATRIX
      N = length(trips.depot); % number of activities
      AllocMatrix = diag(ones(1,N));   % allocation matrix - rows -> vehicles,
      % columns -> activities

      AvailableTrips = 1:N;

      if (verbose_debug)
            % DISPLAY TRIPS
            disp('LIST OF TRIPS')
            for i = 1:N,
```

```
            print_trip(num2str(i),trips.depot(i),trips.ti(i),trips.tf(i),..
            trips.distance(i))
      end
end

% INITIAL NUMBER OF VEHICLES AND SCHEDULING
[partial_cost, Cost(:,:,realday)] = CostMatrix(AllocMatrix, trips, ...
criteria,weights, Kmax);
if(verbose)
      disp(['---> Initial Cost = ' num2str(partial_cost) ' ...
      Unitary Costs']);
      disp([' Number of vehicles (initial) : ' ...
      num2str(sum(sign(sum(AllocMatrix'))) ) ]); disp(' ');
end

for j = 'A':'F', % for all depots

      trips_current_depot = [];

      for k = 1:length(AvailableTrips), % for all (still) available trips
            if(trips.depot(AvailableTrips(k)) == j)
                  trips_current_depot = [trips_current_depot k];
            end
      end

      if (verbose_debug)
            disp(['Depot ' j])
            disp(trips_current_depot)  % trips for the current depot
      end

      trips_left_in_current_depot = trips_current_depot;

      % for all trips in the depot
      for k = 1:length(trips_current_depot),
            rest_of_trips = [];
            last_time = trips.tf(trips_current_depot(k));

            for l = (k+1):length(trips_left_in_current_depot)
                  % check if the other trips in the depot are compatible
                  % with vehicle
```

```
                        if((AllocMatrix(trips_current_depot(k), ...
                        trips_current_depot(l)) == 0)...
                        &&(AllocMatrix(trips_current_depot(l), ...
                        trips_current_depot(l))~=0) ...
                        && is_compatible_trip(trips, trips_current_depot(k), ...
                        trips_current_depot(l), AllocMatrix))
                              % exchange values
                              AllocMatrix(trips_current_depot(k), ...
                              trips_current_depot(l)) = 1;
                              AllocMatrix(trips_current_depot(l), ...
                              trips_current_depot(l)) = 0;
                        else
                              % if not, create a list with the rest of the trips
                              % to allocate to other vehicles
                              rest_of_trips = [rest_of_trips ...
                              trips_current_depot(l)];
                        end
                  end

                  trips_left_current_depot = rest_of_trips;
            end

      end

      if(verbose)
            disp(['---> Final Cost = ' num2str(CostMatrix(AllocMatrix, ...
            trips, criteria, weights,Kmax)) ' Unitary Costs']);
            disp([' Number of vehicles (final): ' ...
            num2str(sum(sign(sum(AllocMatrix'))) ) ]); disp(' ');
      end

      Table(realday, :) = [ day sum(sum(AllocMatrix)) ...
      sum(sign(sum(AllocMatrix'))) ]';
end

Table
cost_A = [Cost(:);ones((realday-1)*D,1)];  % cost vector

% build A_ineq matrix, b_ineq vector and SUM(K(x_tdk-x_(t-1)dk)<=Z_td

per_time_ineq = [];
```

```
for   aux = 0:(Kmax-1),    % build vector [0 0.. 0 1 1.. (k_max-1) (k_max-1)]
      per_time_ineq = [per_time_ineq aux*ones(1,D)];
end

A_ineq = []; b_ineq = [];

for aux = 1:realday,
      A_ineq = [A_ineq  zeros((aux-1),length(per_time_ineq));
                zeros(1,(aux-1)*length(per_time_ineq))  per_time_ineq];

      b_ineq = [b_ineq; Kmax];
end

A_ineq = [A_ineq zeros(realday,(realday-1)*D)];

per_time_ineq = [];
for aux = 0:(Kmax-1),
    per_time_ineq = [per_time_ineq aux zeros(1,D-1)];
end
for aux=2:realday,
     for aux2=1:D,
           A_ineq= [A_ineq;
            zeros(1,(aux-2)*length(per_time_ineq)+(aux2-1)) -per_time_ineq ...
                per_time_ineq(1:(end-aux2+1))      zeros(1,(realday -...
                aux)*length(per_time_ineq))  zeros(1,(aux-2)*D+(aux2-1)) -1...
                zeros(1,(realday-(aux-2)-1)*D -(aux2) )];
           b_ineq = [b_ineq; 0];
     end
end


% build A_eq matrix and b_eq vector
A_eq = [];
b_eq = ones(realday*D,1); % a vector of ones with dimensions [(num_days x
% num_depots) X 1]
per_time_and_depot_eq = [];
for aux = 0:(Kmax-1),
      per_time_and_depot_eq = [per_time_and_depot_eq 1 0 0 0 0 0];
end

for aux = 1:realday,
```

```
      for aux2=1:D,
            new_line = [zeros(1,(aux-1)*Kmax*D) zeros(1,(aux2-1)) ...
            per_time_and_depot_eq(1:(end-aux2+1))  zeros(1,(realday-...
            aux)*Kmax*D) zeros(1,(realday-1)*D)];
            A_eq = [A_eq ; new_line];
      end
end

%-------
options = optimset('Display','iter','BranchStrategy', 'maxinfeas');
init_sol_matrix = zeros( D, Kmax, realday); % compute initial solution
for aux = i:realday,
    for aux2 = 1:D,
      init_sol_matrix(aux2,1,aux) = 1;
    end
end
x0 = init_sol_matrix(:);  % convert the initial solution matrix to vector

LB=0*cost_A;

UB=[ones(D*realday*Kmax,1); Kmax*ones((realday-1)*D,1) ];

x_sol = intlinprog(cost_A',1:length(cost_A),A_ineq,b_ineq, A_eq,b_eq, LB,UB,...
options);

toc

disp(['Final Cost: ' num2str(x_sol'*cost_A)])

MatrixSolution = reshape(x_sol(1:(D*Kmax*realday)), D, Kmax, realday);

disp(MatrixSolution);
```

File “CostMatrix.m”

```
function [output, cost_mtrx_accum] = CostMatrix(AllocMatrix, trips_list, ...
criteria, weights, Kmax)

if nargin < 3,
```

```
    criteria = 1; % cost = trip done or not
end
vehicles_vector = sum(AllocMatrix');      % number of trips per vehicle
ind_used_vehicles = find(vehicles_vector > 0); % index of used vehicles

% get vehicles per depot from the ind_used_vehicles
index_depot = 0;
for j = 'A':'F',
    index_depot = index_depot+1;
    vehicles_depot{index_depot} = [];
    for i = 1:length(ind_used_vehicles)

            if(trips_list.depot(ind_used_vehicles(i)) == j)
                  vehicles_depot{index_depot} = [vehicles_depot{index_depot} ...
                  ind_used_vehicles(i)];
            end

    end
    vehicles_depot{index_depot};
end

% get cost of trips by depot
for i=1:6, % 6 - number of depots
   cost_depot{i} = {};
end

for i = 1:length(vehicles_depot),
    cost_depot{i} = zeros(1,length(vehicles_depot{i}));
    for k = 1:length(vehicles_depot{i})
        ind_trips  = find(AllocMatrix(vehicles_depot{i}(k),:) > 0);
        cost_depot{i}(k) = 0;
        for j = 1:length(ind_trips),
            cost_depot{i}(k) = cost_depot{i}(k) + ...
            cost_trip(trips_list,ind_trips(j), criteria, weights);
        end
    end
    [cost_sorted_depot{i}, Ind_depot{i}] = sort(cost_depot{i},'ascend');
     cumsum_cost_sorted_depot{i} = cumsum(cost_sorted_depot{i});
end

cost_mtrx_accum = zeros(6,Kmax);% number of depots x Kmax matrix to be returned
```

```
for i=1:6,
      if(Kmax >= length(cost_depot{i})),
            cost_mtrx_accum(i,1:length(cost_depot{i})) = ...
            cumsum_cost_sorted_depot{i}(end:-1:1);
      else
            cost_mtrx_accum(i,1:Kmax) = ...
            cumsum_cost_sorted_depot{i}(end:-1:(length(cost_depot{i})-Kmax+1));
      end
end

output = sum(cost_mtrx_accum(:,1)); % sum();
```

File “cost_trip.m”

```
function output = cost_trip(trips_list, index_trip, criteria, weights)

switch criteria
    case 1
            output = 1;

    case 2
            output = trips_list.tf(index_trip)-trips_list.ti(index_trip);

    case 3
            output = weights*[trips_list.tf(index_trip) - ...
            trips_list.ti(index_trip) trips_list.distance(index_trip)]';

    otherwise
            output = 1;
end
```

4.5 *I risultati*

I seguenti risultati sono stati ottenuti su tre differenti istanze di prova (tre giorni di prenotazione), considerando come costo di un viaggio la sua durata, data dalla differenza tra l’ora di arrivo e l’ora di partenza.

```
Kmax = 40   Number of depots = 6  Processing days 1-3 Oct 2014
Criteria = 2
============================================================

Optimizing vehicle usage per day, without restrictions
============================================================

Start processing
trips011014.txt
---> Initial Cost = 218.75 Unitary Costs
 Number of vehicles (initial) : 88
 Number of vehicles (final): 38

trips021014.txt
---> Initial Cost = 224 Unitary Costs
 Number of vehicles (initial) : 83
 Number of vehicles (final): 42

trips031014.txt
---> Initial Cost = 248.5 Unitary Costs
 Number of vehicles (initial): 95
 Number of vehicles (final): 46


Total initial cost = 691.25

Initial table (Day of Month, N trips, N cars used)
====================================================

Table =

     1    88    38
     2    83    42
     3    95    46


Solving Integer Linear Programming with Kmax = 40
====================================================

LP: Optimal objective value is 18.000000.
```

```
Optimal solution found.

Intlinprog stopped at the root node because the
objective value is within a gap tolerance of the optimal value;
options.TolGapAbs = 0 (the default value). The intcon variables are
integer within tolerance, options.TolInteger = 1e-05 (the default
value).

Elapsed time is 1.976587 seconds.

Final Cost: 18

Matrix solution

(:,:,1) =

  Columns 1 through 7

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     1
     0     0     0     0     0     0     1

  Columns 8 through 14

     1     0     0     0     0     0     0
     0     0     1     0     0     0     0
     0     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 15 through 21

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

```
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 22 through 28

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 29 through 35

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 36 through 40

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0


(:,:,2) =

  Columns 1 through 7

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     1
     0     0     0     0     0     0     1
```

Columns 8 through 14

1	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 15 through 21

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 22 through 28

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 29 through 35

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 36 through 40

```
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

(:,:,3) =

  Columns 1 through 11

     0     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0

  Columns 12 through 22

     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0

  Columns 23 through 33

     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0

  Columns 34 through 40

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Inizialmente vengono visualizzati i dati relativi alla schedulazione dell'algoritmo greedy: la tabella indica i giorni di prenotazione analizzati con i rispettivi viaggi da compiere ed i veicoli minimi necessari a soddisfare le richieste.

Il costo totale iniziale è la somma delle durate dei viaggi in programma nei tre giorni, poiché viene calcolato il costo associato all'utilizzo giornaliero di 0 veicoli in ogni stazione. Il solutore "intlinprog" individua il valore ottimo (pari a 17) senza l'applicazione di metodi per la programmazione lineare intera, poiché la differenza tra l'upper bound ed il lower bound è minore o uguale a 0 (valore della tolleranza assoluta).

Le soluzioni trovate sono intere perché rispettano la deviazione massima entro la quale possono essere considerate intere; inoltre sono visualizzati il tempo di esecuzione dell'algoritmo e la soluzione ottima del problema, costituita da una matrice binaria tridimensionale, nella quale le righe rappresentano i depositi, le colonne le auto utilizzate e la terza dimensione i giorni di prenotazione.

Il vettore z_{td}, il quale rappresenta il numero di auto da movimentare giornalmente in ogni deposito, è parte della soluzione ed è una voce di costo da minimizzare, tuttavia non viene stampato a video perché il suo valore è deducibile dalla matrice (la differenza tra il numero di veicoli utilizzati nel deposito d nel giorno t ed il numero di veicoli nel deposito d nel giorno $t-1$).

Come possiamo notare, i vincoli del modello di ottimizzazione sono rispettati: per ciascun giorno, la somma dei valori k corrispondenti alle colonne contenenti 1 è minore o uguale a $k\max$ (bisogna associare un k ad ogni 1 nella matrice) e la somma degli elementi di ogni riga è esattamente pari ad 1.

Nel primo giorno sono richiesti 38 veicoli, ma ne vengono comunque allocati 40, poiché utilizzarne più di quanti ne sono richiesti non comporta alcuna spesa di mancato servizio ed allo stesso tempo minimizza i costi di spostamento che bisogna sostenere se in un giorno vengono richieste più auto di quante ne sono disponibili il giorno precedente.

Nel secondo e terzo giorno, invece, si sostengono i costi associati agli utenti non serviti $\cos t_{tdk}$, dato che sono necessarie 42 e 46 auto, a fronte delle 40 a disposizione. Valutiamo ora

il comportamento del programma al variare del parametro $k\max$, ponendolo dapprima uguale a 48.

```
Kmax = 48   Number of depots = 6  Processing days 1-3 Oct 2014
Criteria = 2
============================================================

Optimizing vehicle usage per day, without restrictions
============================================================

Start  processing
trips011014.txt
---> Initial Cost = 218.75 Unitary Costs
 Number of vehicles (initial) : 88
 Number of vehicles (final): 38

trips021014.txt
---> Initial Cost = 224 Unitary Costs
 Number of vehicles (initial) : 83
 Number of vehicles (final): 42

trips031014.txt
---> Initial Cost = 248.5 Unitary Costs
 Number of vehicles (initial) : 95
 Number of vehicles (final): 46


Total initial cost = 691.25

Initial table (Day of Month, N trips, N cars used)
====================================================

Table =

     1    88    38
     2    83    42
     3    95    46
```

```
Solving Integer Linear Programming with Kmax = 48
=================================================
LP:                Optimal objective value is 0.000000.

Cut Generation:    Applied 1 clique cut, 2 mir cuts,
                   and 4 strong CG cuts.
                   Lower bound is 0.000000.

Branch and Bound:

nodes              total       num int     integer           relative
explored           time (s)    solution    fval              gap (%)
11                 0.07        1           1.605000e+02      9.938074e+01
19                 0.07        2           1.100000e+01      9.166590e+01
27                 0.07        3           3.000000e+00      7.499812e+01
39                 0.07        4           2.000000e+00      6.666444e+01
55                 0.08        5           1.000000e+00      4.999750e+01
56                 0.08        6           0.000000e+00      0.000000e+00

Optimal solution found.

Intlinprog stopped because the
objective value is within a gap tolerance of the optimal value;
options.TolGapAbs = 0 (the default value). The intcon variables are
integer within tolerance, options.TolInteger = 1e-05 (the default
value).

Elapsed time is 0.730382 seconds.

Final   Cost: 0

Matrix solution

(:,:,1) =

  Columns 1 through 7

         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         1
         0         0         0         0         0         0         0
```

```
     0     0     0     0     0     0     0
     0     0     0     0     0     0     1


Columns 8 through 14

     0     1     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     1     0     0
     1     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 15 through 21

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 22 through 28

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 29 through 35

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

Columns 36 through 42

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 43 through 48

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

(:,:,2) =

Columns 1 through 7

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	1

Columns 8 through 14

0	1	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0

Columns 15 through 21

```
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

Columns 22 through 28

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

Columns 29 through 35

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

Columns 36 through 42

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

Columns 43 through 48

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
     0     0     0     0     0     0

(:,:,3) =

  Columns 1 through 7

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     1

  Columns 8 through 14

     0     1     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     0     0     0
     0     0     1     0     0     0     0
     1     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 15 through 21

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

  Columns 22 through 28

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

```
Columns 29 through 35

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

Columns 36 through 42

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0

Columns 43 through 48

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Il valore ottimo della funzione obiettivo è 0 perché, essendo richiesti ogni giorno meno di 48 veicoli, siamo in grado di effettuare tutti i viaggi senza dover sostenere alcun costo di mancato servizio ($cost_{t\,d\,k} = 0$ per $k \geq v_{t\,d}$) e di movimentazione dei mezzi nelle stazioni.

Possiamo osservare come in questo caso “intlinprog” non si arresti alla risoluzione del rilassamento lineare del problema, ma proceda all’esecuzione del metodo dei piani di taglio (ne vengono applicati 7 di tre tipologie differenti) e del *Branch and Bound* (56 nodi esplorati). Analizziamo infine la situazione in cui si hanno a disposizione soltanto 10 veicoli.

```
Kmax = 10   Number of depots = 6  Processing days 1-3 Oct 2014
Criteria = 2
================================================================

Optimizing vehicle usage per day, without restrictions
============================================================

Start  processing
trips011014.txt
---> Initial Cost = 218.75 Unitary Costs
 Number of vehicles (initial) : 88

---> Final Cost = 218.75 Unitary Costs
 Number of vehicles (final): 38

trips021014.txt
---> Initial Cost = 224 Unitary Costs
 Number of vehicles (initial) : 83

---> Final Cost = 224 Unitary Costs
 Number of vehicles (final): 42

trips031014.txt
---> Initial Cost = 248.5 Unitary Costs
 Number of vehicles (initial) : 95

---> Final Cost = 248.5 Unitary Costs
 Number of vehicles (final): 46


Total initial cost = 691.25

Initial table (Day of Month, N trips, N cars used)
============================================================

Table =

     1    88    38
     2    83    42
     3    95    46
```

```
Solving Integer Linear Programming with Kmax = 10
=================================================
LP: Optimal objective value is 420.000000.


Optimal solution found.

Intlinprog stopped at the root node because the
objective value is within a gap tolerance of the optimal value;
options.TolGapAbs = 0 (the default value). The intcon variables are
integer within tolerance, options.TolInteger = 1e-05 (the default
value).

Elapsed time is 0.640325 seconds.
Final   Cost: 420
Matrix solution

(:,:,1) =

  Columns 1 through 7

         0         1         0         0         0         0         0
         1         0         0         0         0         0         0
         0         0         1         0         0         0         0
         0         0         1         0         0         0         0
         0         0         0         1         0         0         0
         0         0         1         0         0         0         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0


(:,:,2) =
```

```
  Columns 1 through 7

       0     0     1     0     0     0     0
       0     1     0     0     0     0     0
       0     1     0     0     0     0     0
       0     0     1     0     0     0     0
       0     0     1     0     0     0     0
       0     0     1     0     0     0     0

  Columns 8 through 10

       0     0     0
       0     0     0
       0     0     0
       0     0     0
       0     0     0
       0     0     0

(:,:,3) =

  Columns 1 through 7

       0     0     1     0     0     0     0
       0     1     0     0     0     0     0
       0     1     0     0     0     0     0
       0     0     1     0     0     0     0
       0     0     1     0     0     0     0
       0     0     1     0     0     0     0

  Columns 8 through 10
       0     0     0
       0     0     0
       0     0     0
       0     0     0
       0     0     0
       0     0     0
```

Come possiamo notare, il costo finale è piuttosto elevato a causa del limitato numero di veicoli disponibili. Oltre al mancato servizio, dalla matrice risulta che bisogna sostenere anche il costo di spostamento di due auto: il secondo giorno nella prima stazione (prima riga) sono richiesti due veicoli, ma nel primo giorno ne viene utilizzato soltanto uno, dunque è necessario movimentare un mezzo; lo stesso avviene nel secondo deposito, infatti il secondo giorno è richiesta un'auto, mentre nel primo giorno non ne viene allocata alcuna.

Chiaramente meno risorse si hanno e maggiore è la probabilità di dover movimentare i veicoli tra le stazioni, al fine di renderli disponibili laddove i costi di mancato servizio sono più alti.

Se volessimo visualizzare $z_{t\,d}$ come vettore, possiamo osservare direttamente il numero di auto da spostare nei rispettivi depositi:

```
>> x_sol((end-realday*D):end)

ans =

     0
     0
     0
     0
     0
     0
     0
     1
     1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

Conclusioni

Il modello di ottimizzazione proposto è finalizzato al miglioramento dell'efficienza di gestione del car sharing elettrico dell'Università di Bologna. Attualmente le auto sono dislocate nelle stazioni secondo un criterio prestabilito che è frutto di una stima della potenziale densità di domanda in ciascuna isola, ma un tale approccio limita la possibilità di raggiungere un buon livello di servizio.

Al crescere delle prenotazioni, infatti, molti utenti non possono effettuare i viaggi richiesti a causa della staticità della flotta, la quale dovrebbe invece essere movimentata a seconda delle esigenze. Questa modalità di gestione richiede certamente degli sforzi organizzativi aggiuntivi, tuttavia consente di massimizzare l'utilizzo delle auto e di valutare il numero ottimo di risorse da allocare giornalmente in ogni deposito.

Nei risultati mostrati la strategicità di ciascun veicolo è misurata in termini di durata totale dei viaggi assegnati, ma i criteri possono essere molteplici, tra i quali il numero di utenti serviti, la distanza percorsa o una combinazione di tali fattori.

Il modello di ottimizzazione è applicabile non soltanto al caso dell'Università di Bologna, ma, variando opportunamente i parametri (dimensione della flotta, stazioni, giorni di prenotazione da analizzare), può essere esteso a qualsiasi altro servizio di car sharing elettrico. La mancanza di una totale operatività dei veicoli può causare ripercussioni negative sui livelli di produttività ed un modo efficace per raggiungerla risiede in una puntuale attività di pianificazione, tesa a configurare il servizio in base alle effettive risorse disponibili. L'obiettivo di questo lavoro è dunque quello di fornire un contributo al miglioramento dell'efficienza gestionale e della qualità di un sistema di condivisione di auto elettriche.

Per organizzare correttamente la distribuzione del parco macchine, in una fase preliminare risulta comunque necessario identificare gli elementi critici di informazione, come il numero dei clienti potenziali, la loro dislocazione geografica, la tipologia e il livello di utilizzo dei veicoli. Un altro importante aspetto è il collegamento del sistema informatico con le auto, in quanto permette, oltre che l'accesso dell'utente al car sharing, la trasmissione automatica di tutti i dati sul loro stato di carica, al fine di controllarne le condizioni funzionali.

Le esperienze finora sviluppate dimostrano inoltre che l'introduzione di una maggiore automazione nell'erogazione del servizio determina due conseguenze favorevoli correlate tra

loro: la prima concerne la razionalizzazione dei processi operativi, la quale comporta la riduzione dei costi e dei tempi di esercizio; la seconda riguarda un maggiore controllo della qualità del servizio che conduce ad una maggiore soddisfazione degli utenti.

Il car sharing elettrico rappresenta un'importante opportunità a favore di una mobilità ecosostenibile e condivisa, grazie alla quale i cittadini possono avvalersi di un nuovo modo di concepire il trasporto. Il servizio si basa sulla considerazione che una macchina è uno strumento utile che deve essere utilizzato in un modo ben ponderato, efficiente e selettivo, poiché, dopotutto, non sempre è la scelta più conveniente per ogni tipo di spostamento.

La libertà di movimento, conseguente alla diffusione dei mezzi di trasporto privati, è una delle conquiste più importanti dell'uomo contemporaneo. Lo sviluppo e la diffusione delle auto, disponibili per tutte le fasce sociali, ha determinato nel corso dell'ultimo secolo un cambiamento epocale nei costumi e nelle abitudini, tanto da caratterizzare in modo indelebile lo sviluppo della società moderna. È del tutto evidente come oggi non si possa più immaginare un mondo privo di autoveicoli, ma allo stesso tempo non si possono più sottovalutare le gravi conseguenze legate alla loro diffusione massiccia.

Il traffico cittadino, l'inquinamento atmosferico, quello acustico e l'erosione crescente di preziose risorse non rinnovabili, sono solo alcune delle problematiche di cui dobbiamo necessariamente occuparci, con l'obiettivo, ormai irrinunciabile, di riuscire a coniugare le esigenze della vita moderna con la salvaguardia del nostro pianeta.

Bibliografia e sitografia

Adler J.D., *Routing and Scheduling of Electric and Alternative-Fuel Vehicles*, Ph.D. thesis, Arizona State University, May 2014.

Bakker J., *Contesting range anxiety: the role of electric vehicle charging infrastructure in the transportation transition*, alexandria.tue.nl., Eindhoven University of Technology, 2011.

Bertossi A., Carraresi P., Gallo G., *On some matching problems arising in vehicle scheduling models*, in «Networks», 17, 1987, pp. 271-281.

Bianco L., Mingozzi A., Ricciardelli S., *A set partitioning approach to the multiple depot vehicle scheduling problem*, in «Optimization Methods and Software», 3, 1994, pp. 163-194.

Boschian V., Clemente M., Ukovich W., *Car sharing e flotte aziendali: modelli per la gestione*, Dossier dell'Università degli Studi di Trieste, 2012.

Bunte S., Kliewer N., *An overview on vehicle scheduling models*, in «Public Transp.», 1, 2009, pp. 299-317.

Fischetti M., Toth P., *Heuristic algorithms for the multiple depot vehicle scheduling problem*, in «Management Sci.», 39, 1993, 115-125.

Freling R., Wage lmans A.P.M., Paixão J.M.P., *Models and algorithms for vehicle scheduling*, in «Transp. Sci.», 35, 2001, pp. 165-180.

Linee guida per l'utilizzo delle autovetture del progetto "Zeta A" come mezzi di servizio condivisi da parte del personale universitario, Regolamento dell'Università degli Studi di Bologna, 2014.

Sovacool B.K., Hirsh R.F., *Beyond batteries: an examination of the benefits and barriers to plug-in hybrid electric vehicles (PHEVs) and a vehicle-to-grid (V2G) transition*, in «Energy Policy», 37, 2009, pp. 1095-1103.

Valenti G., Mastretta M., *Car sharing: evoluzione e prospettive*, documento web, 2002.

Wang H., Shen J., *Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints*, in «Appl. Math. Comput.», 190, 2007, pp. 1237-1249.

www.bmw.it

www.car2go.com

www.citroen.it

www.civitas.eu

www.e-station.it

www.edison.it

www.enjoy.eni.com

www.euromobility.org
www.magazine.unibo.it
www.mathworks.com
www.renault.it
www.sharingitalia.it
www.teslamotors.com
www.tper.it
www.wired.it