Un'applicazione di Smart Parking sociale

Sistemi Multi-agente

A.A. 2014/15

Luigi Bove
N97/196

Luca Capezzuto
N97/205

Marco Valentino
N97/206

1 Introduzione

Lo *Smart Parking* è uno dei punti fondamentali nell'ambito delle Smart Cities in quanto, nell'obiettivo generale di migliorare la qualità della vita in città, una Smart City ha bisogno di soluzioni intelligenti e sostenibili al problema dei parcheggi. Cercare parcheggio infatti congestiona il traffico, aumenta l'inquinamento atmosferico, ed è anche molto frustrante per gli automobilisti, senza contare che maggiore è la densità di popolazione, maggiore è l'entità del problema [1].

Nella ricerca di una soluzione, sono state avanzate diverse proposte che fanno uso di sensori e analisi dei dati, a cui il cittadino si interfaccia tramite applicazioni mobile. Molti degli approcci proposti interpretano lo Smart Parking come un problema di ottimizzazione dal punto di vista degli automobilisti. Tuttavia, un'applicazione di Smart City dovrebbe beneficiare anche la città stessa: oltre ad aiutare gli automobilisti, bisognerebbe anche ottimizzare lo stato dei parcheggi. Altrimenti, trovare parcheggio diviene una "competizione" tra automobilisti, con la conseguenza che chi "perde" è costretto ad effettuare un'altra ricerca, portando ad altre congestioni e altro inquinamento.

In questo documento presentiamo un'applicazione che fa uso di tale approccio. Il progetto rappresenta lo sviluppo e l'ampliamento del prototipo presentato in [2].

2 Il contesto di un'applicazione di Smart Parking

Un sistema di Smart Parking è composto da diversi dispositivi hardware per il rilevamento del grado di occupazione dei parcheggi cittadini, e di applicazioni software per la gestione dell'allocazione dei parcheggi, con i quali gli automobilisti si interfacciano. Di solito tali sistemi permettono all'utente di scegliere da un insieme di opzioni [3].

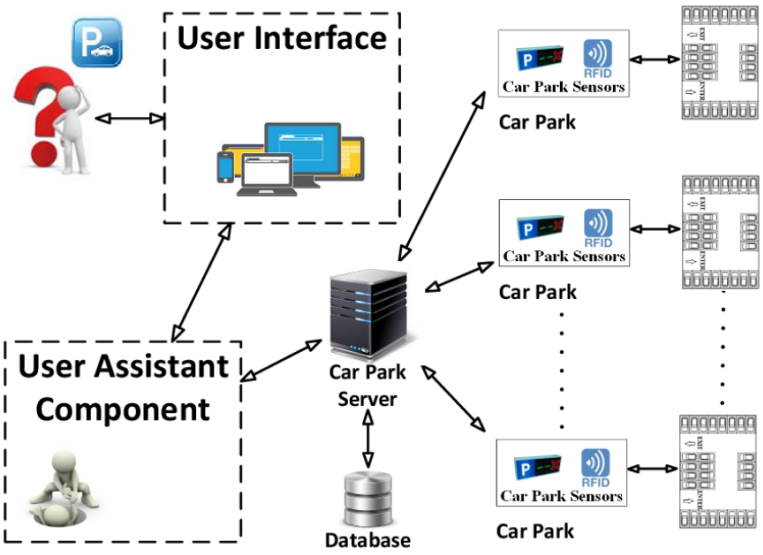

Figura 2.1 Un tipico sistema di Smart Parking. Fonte: [2]

Nel presente lavoro, assumeremo che le Smart Cities siano dotate di un sistema del genere, e proporremo di estenderlo con un modulo software che effettui decisioni per gli automobilisti, considerando non solo i bisogni del singolo, ma anche il beneficio sociale della città. La scelta di un parcheggio sarà il risultato di una negoziazione automatica tra due tipologie di agenti software:

- gli *User Agent* (UA), che agiscono per conto degli automobilisti;

- i *Parking Manager* (PM), che si occupano di gestire i parcheggi cittadini.

È necessario dunque che i diversi proprietari dei parcheggi in città siano favorevoli a sottoscriversi ad un *sistema di parcheggio cittadino*, gestito tramite i Parking Manager, i quali potranno al contempo raccogliere informazioni riguardanti gli specifici bisogni della città, e fornirli agli uffici comunali affinché possano essere valutati.

La negoziazione dovrà trovare il giusto bilancio tra i bisogni del cittadino e quelli della città. Infatti, mentre l'obiettivo del Parking Manager è allocare i parcheggi cercando di ottenere una distribuzione il più possibile uniforme, il cittadino vorrebbe parcheggiare il più vicino possibile alla sua destinazione, e al prezzo minore.

2.1 Un criterio *sociale* per la selezione dei parcheggi

In molti processi decisionali, le alternative competitive e le loro caratteristiche sono note a priori ai decisori. Invece, gli automobilisti trovano i possibili parcheggi in una sequenza temporale, e l'ordine di scoperta influenza la loro decisione. Nel nostro lavoro, il Parking Manager propone i parcheggi in una sequenza temporale che favorisce per primi i parcheggi che più soddisfano i bisogni della città.

Come abbiamo detto, l'obiettivo della negoziazione è trovare un compromesso tra automobilista e città. Questo al fine di un benessere sociale utilitaristico. In altre parole, vogliamo massimizzare il rapporto tra il benessere della società e il benessere del singolo individuo. Il meccanismo di negoziazione proposto si basa infatti sul concetto di *ottimo sociale* [4], secondo il quale tutto ciò che migliora il benessere medio degli agenti che vivono in una società è da considerarsi benefico per la società stessa. Quindi, il benessere sociale che considereremo è la somma delle utilità individuali.

2.1.1 Modello dei prezzi

La possibilità di monitorare la situazione dei parcheggi in tempo reale consente di introdurre un innovativo sistema di prenotazione dei posti basato su prezzi dinamici, come in [1], [5]. In questo modo, gli automobilisti ottengono un servizio migliore, e i gestori, in base a diverse strategie di prezzo, possono ottenere entrate molto più proficue. Ad esempio, le tariffe potrebbero essere più alte durante gli orari di punta, o nelle zone più richieste, e potrebbero esserci sconti per utenti abituali o per chi parcheggia in aree specifiche. Inoltre, in questo modo si potrebbero incoraggiare gli automobilisti a servirsi maggiormente delle strutture park-and-ride[1], aumentando dunque l'uso dei trasporti pubblici e riducendo il traffico cittadino.

Ovviamente, le tariffe andrebbero accuratamente studiate nel contesto specifico di ogni città. In questo lavoro, il Parking Manager cerca di incentivare gli automobilisti a parcheggiare lontano dalle zone altamente congestionate, o in cui ci sono eventi che aumentano il traffico, come concerti, manifestazioni sportive, lavori stradali, e così via. L'uso della negoziazione permette di considerare anche queste informazioni dinamiche.

Per evitare l'agglomeramento di parcheggi in specifiche aree cittadine, il sistema di parcheggio cittadino dovrà dunque utilizzare un modello dei prezzi dinamico. Una volta che il Parking Manager individua una zona in cui si vorrebbe evitare di parcheggiare, questa viene detta *zona rossa*, mentre l'area attorno viene suddivisa in diversi anelli concentrici, che vengono detti *settori*, ed hanno un raggio stabilito secondo un qualche criterio. I prezzi associati ai parcheggi dipendono

[1] Speciali parcheggi connessi ai trasporti pubblici che consentono di raggiungere i centri urbani lasciando il proprio mezzo e usando treni, bus o il car sharing per il resto del viaggio. In genere sono situati nelle zone periferiche e vengono molto usati dai pendolari.

dal settore in cui si trovano, per cui più il settore è lontano, più il prezzo è basso. Inoltre, per incentivare il parcheggio nelle zone meno occupate, un fattore di sconto viene applicato in base al rapporto tra occupazione e capacità totale della zona.

2.1.2 Il meccanismo di negoziazione

In questo lavoro adottiamo il meccanismo di negoziazione riportato in [6], che si basa sul protocollo FIPA Iterated Contract Net, il quale imita un processo di negoziazione tramite reiterazione di offerte. Il protocollo è organizzato in turni, ciascuno composto da interazioni tra lo UA, che inizia la negoziazione, e il PM, che propone le offerte.

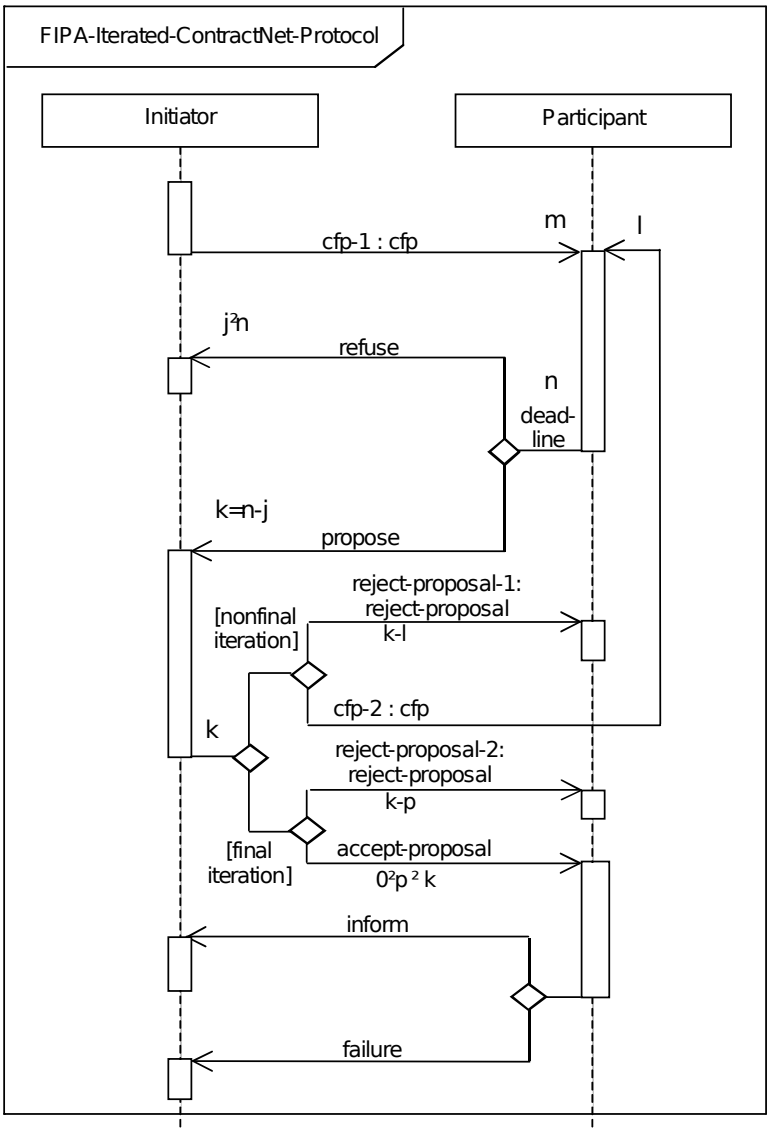

Figura 2.2 Il protocollo d'interazione FIPA Iterated Contract Net. Fonte: [7]

Nel primo turno di negoziazione, lo UA effettua una richiesta (*call for proposal*) di parcheggio, specificando la sua destinazione, l'intervallo temporale in cui ne ha bisogno, e il livello d'importanza (*peso*) circa specifici attributi del parcheggio. Il PM può respingere la richiesta, se non ci sono

offerte disponibili, oppure selezionare un'offerta da una lista di quelle possibili. Nel secondo caso, il PM definisce l'insieme delle offerte selezionando i parcheggi in un'area centrata attorno alla destinazione dell'utente, e le cui dimensioni rispettano un qualche criterio (ad esempio, un'area piccola se l'utente vuole parcheggiare lontano dalla zona rossa, un'area più grande altrimenti). Una volta stabilito l'insieme, il PM calcola i prezzi corrispondenti alle offerte in base al modello dei prezzi descritto nella sezione precedente, poi ordina gli elementi in base a criteri interni che considerano i bisogni della città. Le offerte vengono inviate allo UA, turno dopo turno, in ordine decrescente. Quando riceve un'offerta, lo UA la valuta secondo il proprio criterio e decide se accettarla o rifiutarla. Nel caso in cui rifiuta, il PM invia un'altra offerta, e così via fin quando lo UA accetta o l'insieme di offerte si esaurisce. Una volta rifiutata un'offerta, questa non è più disponibile nei turni di negoziazione successivi. Questa assunzione modella la possibilità che un'offerta rifiutata possa essere riproposta ad un altro utente, o che il suo prezzo possa nel frattempo cambiare secondo l'andamento del mercato, come in [8]. Chiaramente, è difficile per un agente valutare se accettare un'offerta per minimizzare i costi di negoziazione (con il rischio di ottenere un risultato sub-ottimo) o di continuare a valutare offerte per massimizzare la sua utilità attesa (con il rischio di aumentare il costo di negoziazione e arrivare ad un conflitto di interessi). Nel nostro approccio, questo aspetto è modellato associando allo UA una *soglia di accettazione* (nell'intervallo $[0, 1]$), rappresentante l'attitudine dell'utente a raggiungere un compromesso.

Sia i criteri di preferenza del PM che quelli dello UA sono espressi da funzioni basate sull'indice di utilità multi-attributo definito in [9]. In particolare, per entrambi gli agenti l'utilità è definita dalla seguente somma pesata:

$$U_{PM/UA}(\text{offerta}_{PM}(k)) = \sum_{i=1}^{n} \left(w_i * \frac{\text{attr}_i}{\text{fatt}_i} \right)$$

Dove attr_i indica ciascuno degli n attributi considerati per il parcheggio, fatt_i è il corrispondente fattore di normalizzazione, e w_i è il corrispondente peso, con $\sum_{i=1}^{n} w_i = 1$.

La funzione di utilità del PM dipende dalla percentuale di occupazione dei parcheggi al momento della richiesta, e dalla distanza del parcheggio dalla zona rossa, normalizzata rispetto alla massima distanza considerata dal PM, la quale determina l'area per la selezione del parcheggio. La funzione di utilità dello UA dipende dal prezzo del parcheggio, dalla distanza a piedi dal parcheggio alla destinazione, e dal corrispondente tempo di viaggio con i mezzi pubblici. I valori di questi attributi sono mostrati in ogni offerta che il PM invia allo UA. In particolare per lo UA, ogni attributo è normalizzato rispetto al costo massimo di parcheggio e alla distanza massima a piedi dal parcheggio alla destinazione, che sono entrambi specificati come requisiti dall'utente al momento della richiesta [10].

3 Implementazione

Al fine di creare una soluzione che si potesse integrare in un più complesso sistema per la gestione dei parcheggi cittadini, abbiamo implementato un servizio web multi-agente che automaticamente seleziona posti parcheggio in risposta alle richieste degli utenti. L'architettura è mostrata in figura 3.1. Il modulo per la negoziazione, comprendente sia lo UA e il PM che il protocollo di interazione, è stato implementato usando JADE [11], un framework open source per lo sviluppo di applicazioni che implementano sistemi multi-agenti. JADE è basato su Java, e supporta la comunicazione tra agenti in conformità con le specifiche FIPA[2]. Il sistema multi-agente è composto da molteplici UA che comunicano con alcuni PM. Gli UA, insieme ai PM, sono implementati lato server, e

[2] *Foundation for Intelligent Physical Agents.*

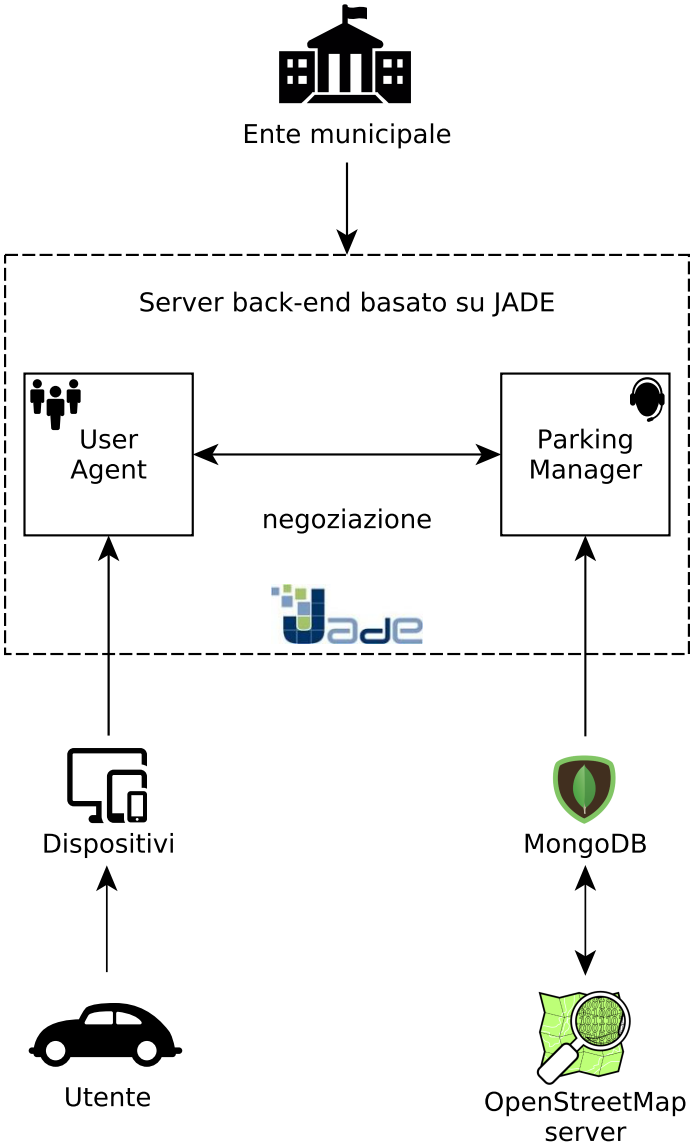

Figura 3.1 L'architettura del servizio.

interfacciati tramite servizi di tipo REST, in modo da ridurre il carico computazionale sui client e centralizzare la gestione dei dati, così che tramite l'esposizione di servizi web sia possibile creare diversi tipi di applicazioni che si interfacciano con l'utente. Il server back-end, basato su Apache Tomcat, è capace di comunicare con diversi servizi e fonti d'informazioni esterne:

- Google Maps [12] per il recupero della distanza a piedi e del tempo di viaggio dai parcheggi alle destinazioni;
- database di parcheggi per conoscere la disponibilità dei posti;
- strutture municipali per il recupero di informazioni riguardanti la viabilità stradale.

Il database di parcheggi viene estratto da OpenStreetMap [13], ed è implementato usando MongoDB, una soluzione NoSQL che supporta una grande varietà di dati, in particolare quelli geografici. La prenotazione del parcheggio, e dunque il decremento dei posti disponibili, viene effettuata solo

nei momento in cui l'utente accetta un'offerta, per non influenzare la funzione di utilità degli agenti. In questo modo si evitano alterazioni del processo di negoziazione.

L'utente, tramite il proprio client[3], dovrà fornire al server, oltre ai dati iniziali (partenza, destinazione, prezzo massimo), i pesi da dare agli attributi per la funzione di utilità della negoziazione. Dopodiché ci sarà una comunicazione in più fasi:

- il client trasforma, tramite un servizio di geocoding, gli indirizzi di partenza e arrivo in coordinate;
- il client compila un JSON per effettuare una HTTP/POST al server;
- il server accetta la richiesta e avvia una negoziazione;
- il client effettua un polling fino a che non ottiene i risultati della negoziazione, che poi elabora per presentarli all'utente (percorso su mappa e indicazioni).

Il polling avviene perché gli agenti vengono eseguiti asincronamente sul server, fino a che la negoziazione non termina e determinate strutture dati vengono modificate con i risultati. La figura 3.2 mostra come avviene la comunicazione tra server e client.

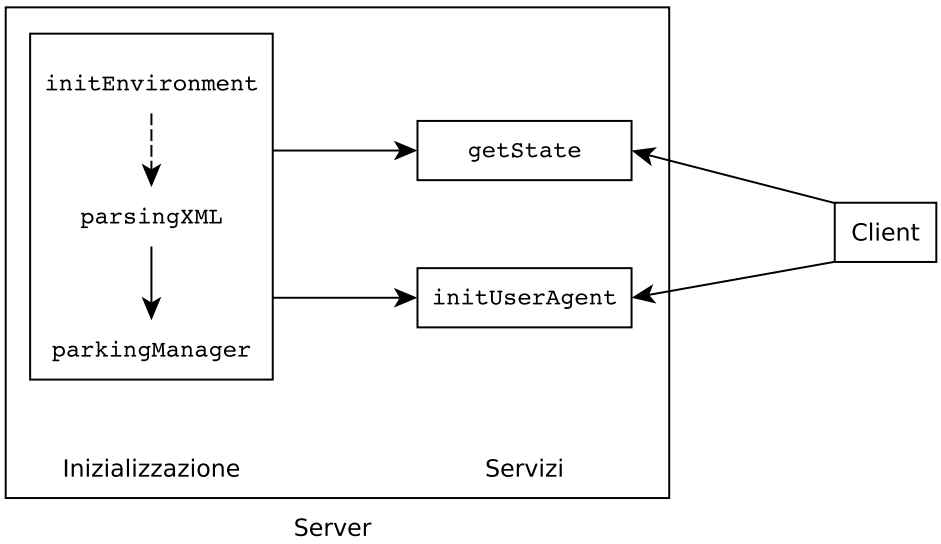

Figura 3.2 Comunicazione tra client e server.

Per prima cosa, il server viene inizializzato tramite tre operazioni:

1. `initEnvironment`, avvio dell'ambiente JADE;
2. `parsingXML`, recupero dei dati XML sui parcheggi, trasformazione in JSON e inserimento in MongoDB: eseguita una prima volta, questa operazione diventa opzionale;
3. `parkingManager`, instanziazione dei Parking Manager, il cui numero dipende dalla mole di connessioni attese.

A questo punto, un client può instanziare uno User Agent invocando il servizio `initUserAgent`. La richiesta necessita di un JSON in input comprendente: partenza, destinazione, prezzo massimo, distanza massima a piedi dal parcheggio alla destinazione, tempo massimo di percorrenza, e soglia di accettazione. Il risultato sarà un URL del genere:

[3] In genere un'applicazione mobile.

```
http://$HOST/SmartParkingServer/initUserAgent?requestJson=$JSON
```

Dove `$HOST` sarà l'indirizzo web su cui risiederà il server, e `$JSON` avrà una forma di questo tipo:

```
[{
        ''partenza'': [40.851816, 14.271955],
        ''arrivo''  : [40.843768, 14.240912],
        ''prezzo''  : 0.145047,
        ''distanza'': 0.0589623,
        ''tempo''   : 0.163915,
        ''soglia''  : 0.0871394
}]
```

Sia `partenza` che `arrivo` sono JsonArray di coordinate. Si noti che lo stesso `$JSON` è in realtà un JsonArray: questo per consentire in implementazioni future la possibilità di introdurre un modulo per effettuare prenotazioni molteplici, e gestire una lista di "preferiti" per ogni utente (ad esempio: parcheggio per il lavoro, parcheggio per andare allo stadio, e così via). Nell'implementazione corrente, viene considerato solo il primo Json.

Effettuata la richiesta, il server risponderà al client con l'identificativo della negoziazione così avviata. Il client userà questo identificativo come input a `getState` per effettuare il polling sullo stato della negoziazione.

Una caratteristica fondamentale del meccanismo di negoziazione implementato è la dinamicità dei prezzi. A seconda del numero di zona, che va da 1 (zona rossa), a 4 (parcheggi più distanti), oltre che dell'occupazione dei parcheggi e dell'eventuale presenza di eventi, più l'utente è meno disposto a parcheggiare lontano dalla zona rossa, più il prezzo lieviterà.

Inoltre, per effetto dell'asincronismo del sistema può capitare che durante una negoziazione i valori di utilità dei parcheggi cambino, perché ad esempio nel frattempo sono stati allocati dei posti nella stessa zona. Per evitare che l'utente si ritrovi ad accettare una proposta che non corrisponde più alla sua utilità attesa, viene usato il valore `soglia` come margine di tolleranza. Dunque, al termine di una negoziazione, se il parcheggio selezionato ha ancora un'utilità maggiore o uguale di `soglia`, allora viene accettato. Altrimenti, lo UA rifiuta tutte le proposte dei PM, ed un nuovo turno di negoziazione viene avviato.

3.1 Caso di test: numerose richieste, stessa destinazione

Il nostro scenario di riferimento consiste in un elevato numero di utenti che effettuano la stessa richiesta nello stesso periodo di tempo. Poiché vogliamo verificare come l'allocazione dei parcheggi viene distribuita tra le zone, non considereremo i fattori di distanza e tempo di percorrenza tra partenza e destinazione. L'obiettivo è evitare il più possibile la zona rossa, così da limitare la congestione del traffico. Il raggio delle zone può essere impostato in base a diversi fattori, come la viabilità corrente, il numero dei parcheggi disponibili, e così via; nel nostro test, ogni raggio di zona è circa 1 km più grande rispetto al precedente. L'esperimento simula 200 richieste effettuate in sequenza nell'arco di 2 minuti, per lo stesso parcheggio a Napoli, quartiere Vomero. Ogni richiesta ha valori di `prezzo`, `distanza` e `soglia` posti a 0.5, `tempo` posto a 0, `partenza` uguale a `arrivo`. I Parking Manager sono tre: NapoliPark, ParkingPrisca, ParcheggiCampania. Le negoziazioni che vanno a buon fine sono le prime 182, in quanto per i rimanenti agenti le utilità scendono al di sotto della soglia di accettazione.

La tabella 1 mostra i valori medi ottenuti durante le negoziazioni. Come si vede, più il parcheggio selezionato è lontano dalla zona rossa, più le utilità degli agenti crescono, e di conseguenza

anche il benessere sociale, ovvero la somma delle utilità individuali, cresce. Ovviamente, il prezzo decresce all'aumentare del numero di zona.

	Prezzo (€)	Utilità PM	Utilità UA	Benessere sociale
zona 1	6.5	0.444	0.531	0.975
zona 2	4.7	0.561	0.618	1.180
zona 3	3.4	0.656	0.685	1.340
totale	4.9	0.553	0.611	1.165

Tabella 1 Valori medi del test.

In tabella 2 è invece mostrata la distribuzione delle allocazioni. Le concentrazioni maggiori si trovano nelle zone più distanti. Questo avviene perché più ci si vuole avvicinare alla zona rossa, più è difficile trovare un compromesso tra gli agenti. All'inizio i parcheggi vengono allocati il più possibile lontani dalla zona rossa, e solo man mano che si riempiono (e quindi la loro utilità cala) vengono scartati in favore di zone più vicine.

Possiamo dunque constatare che i parcheggi vengono scelti in accordo con il nostro obiettivo.

Parcheggio	Manager	Riempimento (%)	Zona
Carrefour, Via Giuseppe Orsi	ParkingPrisca	92.6	3
Via Francesco Solimena	NapoliPark	84.3	3
Parco Montedonzelli	ParcheggiCampania	83.8	3
Autovomero, Via Gianlorenzo Bernini	NapoliPark	75.6	2
Carrefour, Via Antonio Solario	ParcheggiCampania	75	2
Salita Arenella	NapoliPark	75	2
Via Enrico Alvino	ParkingPrisca	72.5	1

Tabella 2 Parcheggi coinvolti nel test.

4 Conclusioni

L'obiettivo dello Smart Parking è rendere più facile agli automobilisti trovare parcheggio, e al contempo favorire una riduzione di traffico, emissioni di CO, frustrazione e perdita di tempo, andando così a beneficiare la città intera. Ma per raggiungere pienamente questo obiettivo, le Smart Cities dovrebbero poter fare affidamento sull'autorità cittadina, in modo da conoscere regolamentazioni e decisioni che aiuterebbero a sviluppare servizi migliori. Tipicamente, queste informazioni sono molto dinamiche e influenzate da eventi imprevisti. Per questo motivo, abbiamo proposto un'applicazione di Smart Parking che punta ad un beneficio sociale piuttosto che solo del singolo. Il meccanismo implementato punta a massimizzare il benessere collettivo cercando un compromesso tra utenti e autorità cittadine, i cui interessi sono spesso in conflitto. Il modo classico per ottenere ciò è tramite una negoziazione, che è il punto chiave della nostra applicazione, e l'oggetto su cui si è basato questo documento.

Riferimenti

[1] Polycarpou E., Lambrinos L., Protopapadakis E. Smart parking solutions for urban areas. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE 14th International

Symposium and Workshops on a, June 2013.

[2] Di Nocera D., Di Napoli C., Rossi S. A social-aware smart parking application. In *Proceedings of the 15th Workshop "From Objects to Agents", Catania (Italy)*, volume 1260, September 2014.

[3] Faheem Z., Mahmud S., Khan G., Rahman M. A survey of intelligent car parking system. *Journal of Applied Research Technology*, 11(5):714–726, 2013.

[4] Arrow K., Sen A., Suzumura K. *Handbook of Social Choice & Welfare*. Handbooks in Economics. Elsevier Science, 2010.

[5] Meir R., Chen Y., Feldman M. Efficient parking allocation as online bipartite matching with posted prices. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, pages 303–310, 2013.

[6] Di Nocera D., Di Napoli C., Rossi S. Agent negotiation for different needs in smart parking allocation. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems*, volume 8473, pages 98–109. Springer International Publishing, 2014.

[7] FIPA Specifications. Iterated Contract Net Iteration Protocol. http://www.fipa.org/specs/fipa00030, 2002. Accessed: 19-10-2015.

[8] PeterR. Lewis, Paul Marrow, and Xin Yao. Resource allocation in decentralised computational systems: an evolutionary market-based approach. *Autonomous Agents and Multi-Agent Systems*, 21(2):143–171, 2010.

[9] Barbuceanu M., Lo Wai-Kau. Multi-attribute utility theoretic negotiation for electronic commerce. In *Agent-Mediated Electronic Commerce III*, volume 2003 of *Lecture Notes in Computer Science*, pages 15–30. Springer Berlin Heidelberg, 2001.

[10] Mejri N., Ayari M., Kamoun F. An efficient cooperative parking slot assignment solution. In *UBICOMM 2013, The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 119–125, 2013.

[11] Bellifemine F., Caire G., Poggi A., Rimassa G. Jade: A software framework for developing multi-agent applications. lessons learned. *Inf. Softw. Technol.*, 50(1-2):10–21, 2008.

[12] Pan B., Crotts J. C., Muller B. Developing web-based tourist information tools using google map. In *Information and Communication Technologies in Tourism 2007*, pages 503–512. Springer Vienna, 2007.

[13] Haklay M., Weber P. OpenStreetMap: user-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.