Negotiating Parking Spaces in Smart Cities

Claudia Di Napoli
Istituto di Calcolo e Reti ad Alte Prestazioni
C.N.R.
Naples - Italy
claudia.dinapoli@cnr.it

Dario Di Nocera*
Dipartimento di Matematica
University of Naples
"Federico II", Napoli - Italy
dario.dinocera@unina.it

Silvia Rossi
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
University of Naples
"Federico II", Napoli, Italy
silvia.rossi@unina.it

ABSTRACT

Parking in urban areas is becoming a big concern for its environmental and economic implications. Smart parking systems are considered essential to improve both city life in terms of gas emission and air pollution, and motorists life by making it easier to park. Supporting technologies are emerging at the industrial level to easily locate available parking spaces, to automate parking payments, and to collect useful data on consumer demand. Nevertheless, the full potentiality of smart parking systems is still far to come, and it represents a big challenge for the future of Smart Cities. In this paper we propose to address the parking space allocation as the result of an agreement between parking providers and parking requestors that accommodates their respective requirements on some parking attributes. A software agent negotiation mechanism is adopted to establish such an agreement by taking into account user requirements on a parking space in terms of its location and cost, and the vendor requirements in terms of income and city regulations to obtain an efficient parking allocation and traffic redirection. It is shown that agent negotiation allows to allocate parking spaces to users in an automatic and intelligent manner by taking into account that a compromise among different preferences of users and vendors have to be met.

Keywords

Agent negotiation, multi-agent systems, smart parking, smart cities.

1. INTRODUCTION

Urban transportation is considered a relevant investigation area for the innovation of Smart Cities since it may contribute to increase the quality of life of city-dwellers, to enhance the efficiency and competitiveness of the city economy, and to move towards the sustainability of cities by improving resource efficiency and meeting emission reduction targets. The main themes addressed in urban transportation are:

- Cooperative Intelligent Transport Systems and Services (C-ITS), based on the principle that all cooperative parties (i.e. ITS stations, vehicles, road side units) exchange information between each other, so enabling up-to-date traffic information, improved road safety and traffic efficiency.
- Enabling Seamless Multi-modality for End Users, based on the possibility to combine public transport with other motorized and non-motorized modes as well as with new concepts of vehicle ownership.
- Smart Organization of Traffic Flows and Logistics that involves multi-agency interaction, linking individual mobility with public transport services.

In this framework, one of the problems linked to the above themes, is parking in urban areas. It is widely recognized that drivers searching for parking in wide urban areas waste time and fuel, so increasing traffic congestion and air pollution [11]. Most of the research projects concerning smart parking systems focus on ways to collect and publish live parking information to drivers so they can be informed of available parking spaces near to the destination they require [9]. Nevertheless, the fragmentation of public and private parking providers, each one adopting their own technology to collect occupancy data, makes it difficult to advise motorists of available parking in multiple zones, but, more importantly, to help them in making decisions on where to park. Hence, smart parking applications should aim at coordinating individual parking solutions, both private and public, without involving end-users in the fragmentation of parking owners. Individual parking owners should be made aware of the benefits of such a global parking provision by showing them that the coordinated provision of parking solutions still guarantees their individual income and fair competition by better exploiting the parking spaces offered in a city.

In the present work, we investigate the possibility to use software agent negotiation to manage the relationship between parking supply and demand to provide user-oriented automatic parking services that take into account both drivers preferences, and parking vendors requirements together with social benefits for the city, such as a reduction of traffic by limiting parking in city center [13]. We propose to use software agents to model both a Parking Manager, who is responsible for coordinating the offers of individual Parking Owners (both public or private), and motorists who are end users that search for parking spaces that meet their requirements. In particular, an automatic negotiation mechanism

*Ph.D. scholarship funded by Media Motive S.r.l, POR Campania FSE 2007-2013.

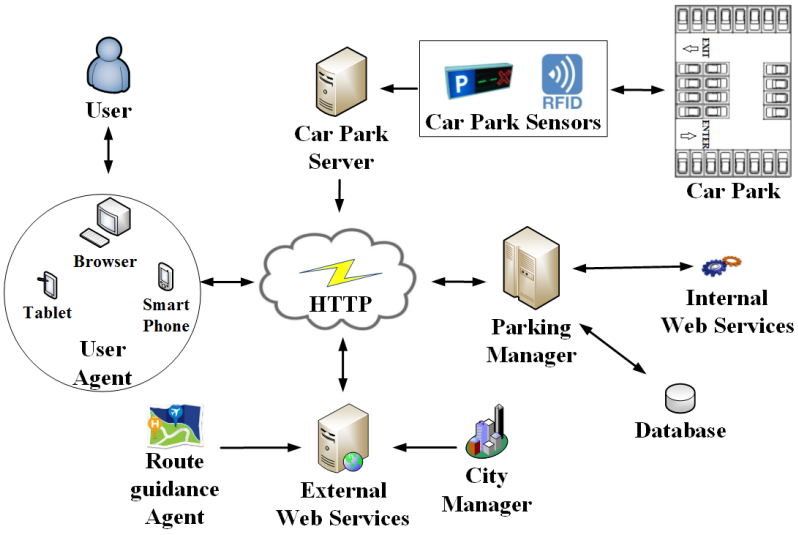


Figure 1: A Car Parking System.

is proposed to accommodate users and providers needs. Of course, the length of negotiation could prevent its use in this setting [5], so it should be adapted to the negotiation trend that may vary because of the the attributes to be negotiated upon, and the parking market situation.

2. A MODEL FOR A CITY PARKING SYSTEM

Car Park Systems refer to a wide spectrum of parking facilities including devices to automatically locate car parks and to automate parking space payment.

In the present work, a Car Park System is intended as a complex application composed of different devices and services, that allows users to retrieve information on the available parking spaces in a city around a specific destination area. A sketch of such a system is reported in Figure 1. As shown, a user may submit a request for a parking space to the Car Park Server through several devices (e.g. Tablet, Smart-Phone, PDA or PC). The system provides the user with a city map to select the area he/she would like to park, and an interface to indicate his/her parking preferences. A Parking Manager (PM) is responsible for processing the request. It queries an internal database (Database) to retrieve information on the available car parks, and it relies on specific applications to extract car park availability at the moment the request is processed (e.g. through Car Park Sensors). Also it may invoke additional services (External Web Services) to collect information on city regulations and/or events (provided under the responsibility of the City Manager) relevant to find a parking space, or other salient information, such as an estimation of the time necessary to arrive to the user destination from a specific car park, that can be retrieved from external applications as Google Maps API [10].

In such a framework, each car park is characterized by the following parameters:

`car_park= <park_id, park_GPS_location, ref_price_unit, park_capacity, sector>`

where `park_id` is the unique identifier of the car park, `park_GPS_location` is its GPS location, `ref_price_unit` is the default time unit price for a parking space, `park_capacity` is the total number of parking spaces of the car park, and `sector` represents the geographical location of the car park with respect to the city center. In fact, in the proposed application, the city is divided in several rings (referred to as *sectors*) that account for the distance between the car park and the city center, as shown in Figure 2. A `sector` is represented by an integer value so calculated:

$$sector = \begin{cases} 0 & distance\_from\_city < min\_range \\ 1 + \left\lfloor log_2(\frac{distance\_from\_city}{min\_range}) \right\rfloor & otherwise \end{cases}$$

where min_range is the radius of the first area (`sector=0`), and $distance\_from\_city$ represents the distance between the car park location and the city center (located in `sector=0`).

A user request (`park_req`) is composed of values referred to the parking space attributes that are relevant for the user to decide where to park.

`park_req(t)= <id_req, dest_GPS_location, start_time, end_time, reserv_time>`

where `id_req` is the unique identifier of the user request, `dest_location` represents the GPS location of the destination the user wants to reach, the time interval (`end_time - start_time`) represents the duration the user wants to park for, and `reserv_time` is a flag used to distinguish between on-demand or advance requests. For the time being, only advance requests are considered since for on-demand requests different assumptions on the evaluation of car park occupancy should be considered.

With a static selection, the PM will select car parks considering only to meet the user requirements in terms of lo-

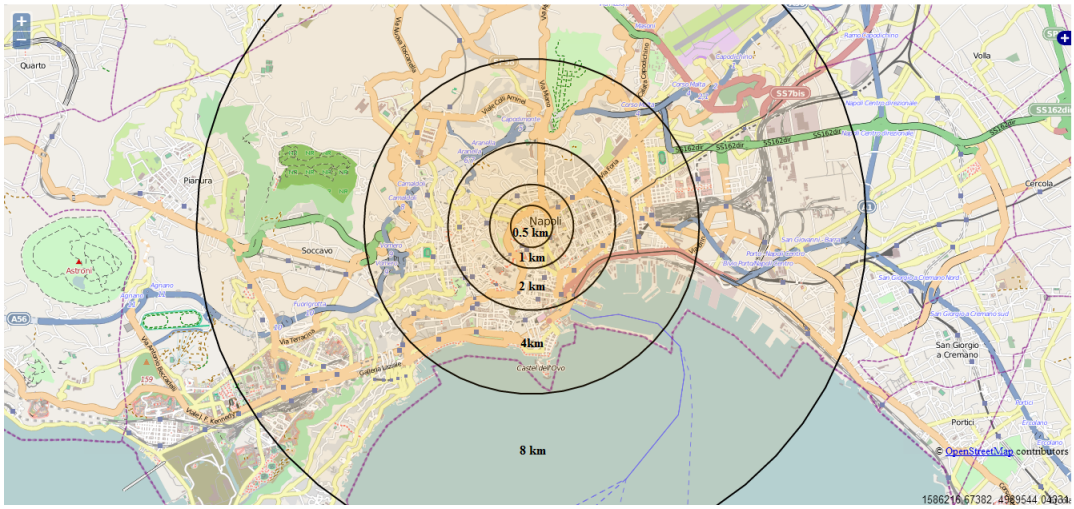


Figure 2: Sector distribution for the city of Naples.

cation, and available parking spaces for the required time interval. If there is no parking space meeting the requirements, a static mechanism will end up with no solutions for the driver request. A dynamic selection of parking spaces implies the evaluation of criteria that may not be explicitly expressed by the user, and that can influence the selection of the parking spaces offered by the PM. Furthermore, users may adopt private evaluation criteria that are specific to their profile to evaluate if the received offer is acceptable or not. With a dynamic selection, parking solutions that were not found with a static selection, could be produced as an acceptable compromise between PM and UA preferences.

3. NEGOTIATING OVER PARKING SPACE ATTRIBUTES

In a smart parking application, motorists will be classified according to their different requirements on parking spaces corresponding to different user's profile (e.g. business, tourist, generic). In fact, users may have different preferences on the parking attributes, and their relative *importance* (measured in terms of *weights*). Furthermore, additional information may be used (that could come from other sources of information) to help refining the selection process, e.g., unavailability of public transportation at the required time, the necessity to reach different locations once the car has been parked, the possibility to find other attractions in the area, and so on.

In this work, we investigate the possibility to use software agent negotiation to provide a user-oriented automatic parking service that takes into account both drivers preferences, and parking vendors requirements together with social benefits for the city. In particular, we propose a negotiation mechanism between two agents: the PM and a User Agent (UA). The PM has the aim to improve the citizen life, and city pollution by decreasing the influx of cars in the city center, and, at the same time, to offer a better distribution of vehicles in the managed car parks, still trying to obtain an economic income. The UA has the aim to help a motorist to select one of the parking solutions proposed by PM. Of course, it is difficult for the negotiating agent to evaluate whether to accept an offer to minimize the expected cost of communication (at the risk of getting a sub-optimal result for the specific application), or to keep on negotiating to maximize its expected utility (at the risk of increasing the cost of negotiation and ending with a conflict deal). Usually this lead to the specification of an acceptance condition that is not only based on utility, but on more complex criteria (i.e., based on utility and time) [2].

The adopted negotiation model is based on the one proposed in [6] that was shown to be a viable approach to address the problem of service selection for Service Based Applications characterized by Quality of Services values that once aggregated should meet user's preferences. The proposed mechanism allows to implement a *flexible* negotiation in terms of its length. In fact, the negotiation proceeds in *rounds*, and the number of round is not statically set, but its value may be changed by the PM or by the UA according to the trend of the negotiation process. A concession strategy is used at each negotiation round by the PM to make offers, and both negotiators may decide to end negotiation according to the negotiation evolution, so the negotiation deadline (i.e. the number of allowed rounds) is not fixed a priori.

3.1 A one-sided negotiation model

Usually negotiation takes place between two agents x and y willing to come to an agreement on conflicting interests, by exchanging an alternate succession of offers and counteroffers in a bilateral interaction [8].

In the present work we adopt the negotiation mechanism reported in [6], whose protocol is based on the Iterated Contract Net Protocol, that is frequently used to mime the human contract negotiation process [4]. Contract net protocol is a market-like mechanism allowing involved parties to exchange information in a distributed system, such as a multi-agent one.

As described in Figure 3, the protocol is organized in *negotiation round*s, each one consisting of interactions between the UA, that is the *initiator* of the negotiation, and the PM, that is the agent proposing offers. Negotiation rounds may be iterated for a variable number of times until a *deadline* is reached or the negotiation is successful. Moreover, both the UA and the PM can stop the negotiation process. At each

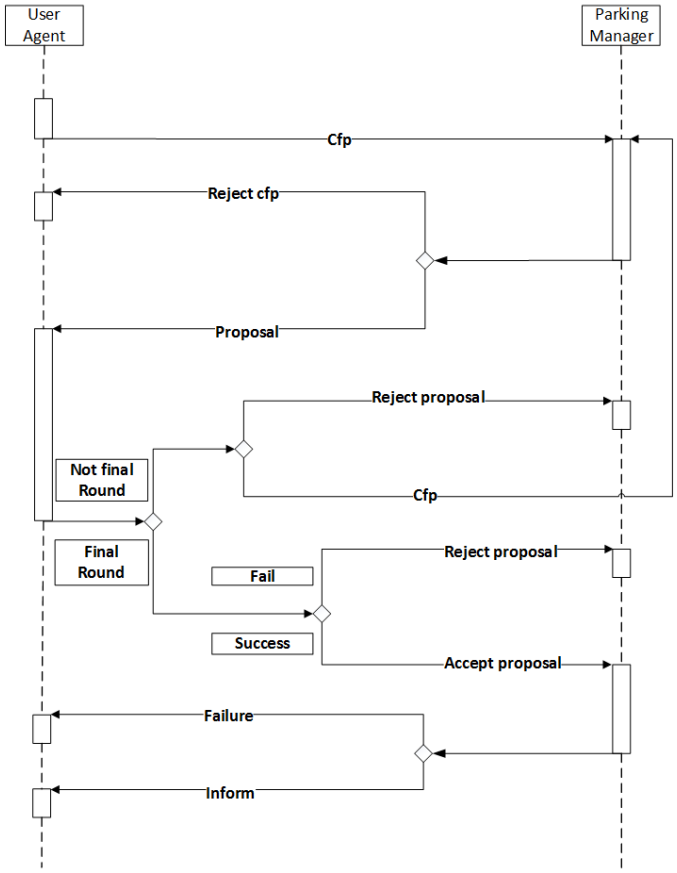


Figure 3: The iterated negotiation protocol.

negotiation round, the UA issues a request for a parking space (`cfp`) specifying its preferred values for the parking attributes; the PM can either reject the call (`Reject cfp`), if there are not offers available, or it sends back a parking solution (`Proposal`) selected from a set of available offers it calculated according to the preferences specified in the `cfp` and its own preference criteria. In the latter case, the UA evaluates the received offer, according to its own evaluation criteria, to decide whether to accept (`Accept proposal`) or to reject it (`Reject proposal`). If the offer is accepted the negotiation ends with an `Inform` message assigning the selected car space to the UA, otherwise a new round starts with the UA sending again the same `cfp` request. It should be noted that an offer proposed by the PM in a negotiation round is not considered available in future rounds once it is rejected. This assumption models the possibility that a rejected parking space may be offered to another user in the meantime, or its price may change according to the parking market trends.

Both PM and UA preferences over the attributes to be negotiated upon, are modeled through utility functions based on the Multi-Attribute Utility Theory defined on independent issues [3]. The function domain represents the *negotiation space*, and it is normalized to the interval $[0, 1]$. So, the utility function of an agent x for an offer o_y sent by the agent y (with $x = y$ or $x \neq y$) is $U_x(o_y) : D_1 \times \cdots \times D^r \rightarrow [0, 1]$, where $D_1, \ldots, D_r$ are the value domains of the r negotiation issues. The utility function allows to evaluate the value of each specific offer in terms of agent utility with respect to that offer.

In our model, the utility function of the PM depends on the car park availability at the moment the request is received, and on the distance of the car park from the city center, while the utility function of the UA depends on the parking space price, and on its distance from the requested destination. Different weights of the different issues may model different classes of UAs and PMs. In this way, the issues considered in the PM utility function take into account the preference of the PM to propose first car parks that are both less occupied and not located in the city center (to reduce the influx of cars in city centers). The issues considered in the UA utility function take into account the preference of the UA concerning the parking space price, and its location with respect to the preferred final destination. Utility functions are modeled as linear functions (as it will be explained in the following sections) resulting from the weighted sum of the considered issues.

The negotiation occurring between the PM and the UA is defined as a *one-sided* negotiation since it allows only the PM to formulate offers, according its own utility function, and the UA only to evaluate them, according its own utility function as well. The rationale of this choice is to model the assumption that UAs do not have complete information on parking spaces availability, otherwise they would simply choose the offer more convenient for them without reaching a compromise also with the preferences of the PM. So, at

each round the PM sends only one offer (or equivalently a finite set of offers) selected according to a strategy allowing to take into account the requirements of both negotiators.

3.2 Parking Manager Behavior

At the first round of negotiation, the PM computes the set of possible offers corresponding to a set of car parks that meet the following requirements:

- the distance (referred to as `park_GPS_distance`) of the car park location (`park_GPS_location`) from the destination (`dest_GPS_location`) set by the user, is within a given distance (`location_tolerance`);
- the car park have spaces available for the time interval specified by the user at the time `t` the request is issued (`end_time - start_time`).

The `location_tolerance` is set by the PM in such a way to include also car parks that are not in the city center, and consequently they may be far from the `dest_GPS_location` specified by the user, since the PM tries to prevent users from parking in the city center and to maximize the occupancy of car parks.

An offer of the PM for a parking space of a selected car park is:

$$\texttt{offer(k)} = < \texttt{park\_id}, \texttt{park\_GPS\_distance}, \texttt{dest\_time\_distance}, \texttt{park\_price\_unit} >$$

where `park_id` is the identifier of the selected car park, `park_GPS_distance` is the distance between `park_GPS_location` and `dest_GPS_location`, `dest_time_distance` is the time necessary to travel from `park_GPS_location` to the `dest_GPS_location` using public transportation, and `park_price_unit` is the unit price offered for the selected parking space. The `dest_time_distance` value is obtained by invoking external services, such as Google Maps, but also also other city services giving additional information such as events preventing the use of public transport at the time of the request.

In order to incentivize users to park outside the city center and in car parks with more parking spaces available, the park unit price for a parking space is dynamically computed by considering that car parks located in the city center are more expensive (according to the ring distribution reported in Figure 1), and that car parks are offered with a discount factor that depends on the car park occupancy. Hence, the `park_price_unit` for a selected car park is computed as follows:

$$\texttt{park\_price\_unit} = \texttt{max\_price}\left(1 - \frac{\texttt{sector}}{\texttt{max\_sector + 1}}\right) + - \frac{\texttt{park\_availability}}{\texttt{park\_capacity}} \cdot u_d$$

where `max_price` is the maximum time unit price for the city center car parks, `max_sector` is the maximum number of sectors in the city, `park_availability` is the number of parking spaces available for the time interval requested by the UA (`end_time - start_time`), `park_capacity` is the total number of parking spaces, and u_d is the maximum discount for the PM on the car parks (with $u_d \ll$ `max_price`). In this way, the price offered by the PM is not the static default price associated to the car park (i.e. `ref_unit_price`), but a dynamic value. The `park_availability` value is retrieved through a specific service invoked by the PM at the time the request is processed.

Once the PM computes the set of possible offers, it needs to establish which one to offer at each negotiation round, i.e. it needs to establish its concession strategy during negotiation. In order to do so, the PM uses a private utility function to rank the selected car parks. The evaluation function used by the PM to compute the utility of an offer ($offer_{PM}(k)$) is the following:

$$U_{PM}(offer_{PM}(k)) = \sum_{i=1}^{n}(\alpha_i * \frac{q_{i,k} - \min_j(q_{i,j})}{\max_j(q_{i,j}) - \min_j(q_{i,j})})$$

where n is the number of issues the agent is evaluating, $q_{i,k}$ is the value of the i-th issue of the k-th car park, and $\min_j(q_{i,j})$ and $\max_j(q_{i,j})$ are respectively the minimum and the maximum values of the i-th issue among all the car parks selected by the PM. The constants α_i are weights associated to different issues with the constraint that:

$$\sum_{i=1}^{n}\alpha_i = 1$$

The issues for the PM are the distance of the car park from the city center, and the availability of parking spaces in the car park for the requested time interval, i.e.:

- $q_1 = dist(\texttt{park\_GPS\_location}, \texttt{center\_GPS\_location})$
- $q_2 = \texttt{park\_availability}$

Through its utility function, the PM ranks the offers for the selected car parks in a utility descending order (total or partial). At each negotiation round, it sends the UA one offer according this order, so adopting a concession strategy with a monotonically decreasing value of utility.

3.3 User Agent Behavior

The UA evaluates the offer it receives at each round to decide whether to accept or to reject it. In order to do so, it calculates its utility value for that specific offer, using the following utility function:

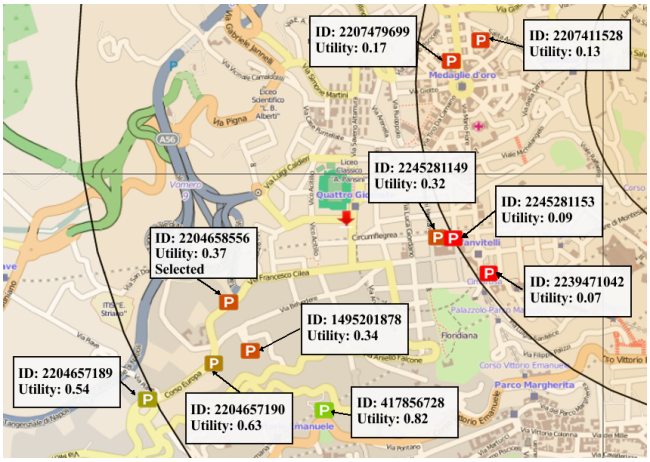
$$U_{UA}(offer_{PM}(k)) = 1 - \sum_{i=1}^{m}\beta_i * \frac{q_{i,k} - c_i}{h_i - c_i}$$

where m is the number of issues the agent is evaluating, $q_{i,k}$ the value i-th issue of the k-th offer, c_i is the preferred value over the i-th issue, and h_i is a constant value introduced for normalizing each term of the formula into the set [0,1]. The constants β_i are weights associates to different issues with the constraint that:

$$\sum_{i=1}^{n}\beta_i = 1$$

If $q_{i,k} - c_i < 0$ than the term $\sum_{i=1}^{m}\beta_i * \frac{q_{i,k}-c_i}{h_i-c_i}$ is set to zero. Moreover, we assume that the preferred c_i values are not unreasonable with respect to each considered issue (i.e. user cannot ask for a parking space in a city center for free!).

The issues considered by the UA are the offered price, the distance of the offered car park from the requested location, and the travel time distance to the offered car park from the requested location with public transportation:



(a) Parking Manager utilities.

(b) User Agent utilities.

Figure 4: Parking Manager and User Agent Utilities.

- q_1 = `park_price_unit`
- q_2 = `park_GPS_distance`
- q_3 = `park_time_distance`

The UA accepts the offer if the utility value for that offer is greater then a predefined *threshold* value. This threshold may be set to different values to model different UA profiles.

4. A FIRST EXPERIMENTATION ON A REAL SETTING

A preliminary set of experiments was carried out to determine whether negotiation is a viable approach in order to meet both users and parking managers requirements.

In this experimentation the weights in the utility functions are equally distributed among issues (i.e., $\alpha_i = 0,5$ and $\beta_i = 0,33$ for all i), while for each issue i, h_i and c_i are dynamically set to respectively $max_j(q_{i,j})$ and $mean_j(q_{i,j})$ (i.e., the maximum and the mean value for the current issue). The UA accepts an offer if its utility for that offer is greater than a threshold value set to 0.6 for all the experiments.

4.1 Utility Evaluation in a Running Example

A running example of a real negotiation, where we evaluate the utility obtained by the PM and the UA when an agreement is achieved, is reported.

The experiment starts with a request issued by a hypothetical user specifying the destination he/she wants to reach, selected on interactive city map provided by a specific service, and the time interval he/she wants to park for. As described in Section 2, the UA sends a `park_req` (i.e., a call for parking) to the PM. A graphical representation of the use case described above is reported in the Figure 4, where the destination selected by the user is identified with the down arrow.

At the first round, the PM selects a list of car parks around the user's destination (as shown in Figure 4(a)), and it calculates the ranking of the selected car parks based on its utility according to the function reported in Section 3.2. The PM found ten car parks with parking spaces available in the requested area within a predefined `location_tolerance`. Parking identifiers and locations are extracted from the OpenStreetMap database [7] of the city of Naples (Italy), while routing information (`dest_GPS_distance` and `dest_time_ distance`) are evaluated through the use of Google MAPs API [12]. The occupancy of car parks is randomly generated for each negotiation run. In the Figure 4(a), the selected car parks are reported with labels specifying the corresponding park ids and their utility values, as evaluated by the PM.

At each negotiation round, the PM offers to the UA the parking space with the highest utility value (in this example it offer a car park with utility equals to 0.82). The UA accepts (rejects) the offer if its utility for that offer, evaluated according to the formula described in Section 3.3, is higher (lower) than the threshold value. The first PM offer corresponds to an utility for the UA equals to 0.33. Hence, the offer is rejected because it is lower than the threshold value (equals to 0.6), and the UA starts another round of negotiation. The negotiation ends at the fourth round, when the UA accepts an offer with utility equals to 0.66 (corresponding to an utility for the PM equals to 0.37). In the Figure 4(b), car parks offered by the PM during negotiation are reported with labels specifying the corresponding park ids and their utility values, as evaluated by the UA.

In Table 1 we summarized all the relevant information at each negotiation round, reporting the number of parking spaces available in a car park (# Spaces), its distance from the city center (Distance), the unit price (Price) to be paid for the parking space, and the distance of the car park from the destination set by the UA, calculated both in length and in time (Route and Time), as obtained by querying a service of Google Maps. This information is necessary to allow the PM and the UA to calculate their utility values for the car parks, according to the utility functions reported respectively in 3.2 and 3.3. In this specific run, the negotiation ends after four rounds with an utility of the PM equals to 0.37 and for the UA equals to 0.66. Note that while the utility of the PM is not particularly high (because of the few parking spaces available in the car parks), the PM still manages to allocate a parking space in only four rounds of

# Rounds	ID	# Spaces	Distance (m)	Price (€)	Route (m)	Time (s)	PM Utility	UA Utility
1°	417856728	109	3187	7.99	1516	1384	0.82	0.34
2°	2204657189	41	4036	5.61	1818	2183	0.63	0.16
3°	2204657190	41	3594	7.98	1192	871	0.54	0.55
4°	2204658556	18	3359	7.46	891	646	0.37	0.66

Table 1: Negotiation on a single query.

negotiation, being able to reach a compromise by offering a car park that is not the closest to the user's destination, but still acceptable by the user in terms of time necessary to reach the destination from the car park location, and that is not too close to the city center.

4.2 1 vs N Rounds of Negotiation

Another experimentation was carried out on a simulation of 150 different queries made by users. The destinations selected by the user are located in sectors two and three on the city map. For each query a negotiation run takes place. The experimental results are summarized in Table 2 for successful negotiations. In particular, the table reports, for each negotiation run, the minimum, the maximum and the mean value (with the standard deviation) of the number of selected car parks (# Available car parks), the number of negotiation rounds (# Rounds), the PM and the UA utility.

The mean value of rounds (that is the mean number of offers sent by the PM) is much lower that the mean number of car parks selected by PM for the experiments (3.3 rounds with respect to 11 available car parks). This means that the negotiation ends before the PM offers all the selected car parks.

The obtained mean utilities values for the UA and PM are reported in rows 3 and 4 of Table 2, showing that a compromise on the requirements of both parties is reached. In fact, without negotiation (i.e., in the case the complete set of offers selected by the PM is known to the UA as well), the UA would select the offer that maximizes its own utility. The PM and the UA mean value utilities without negotiation are reported in the last two rows of Table 2. As expected, in this way, the UA requirements are privileged (UA achieves a mean utility value equals to 0.71) with respect to the PM ones (PM achieves a mean utility value equals to 0.35).

5. DISCUSSION AND CONCLUSIONS

Parking in populated urban areas is becoming a challenging problem requiring smart technologies in order to assist users in finding parking solutions, and to shorten the time necessary to find parking spaces. In this way, it is possible to decrease traffic congestion, and to improve the everyday life of city dwellers.

In the present work, we investigated the possibility to use software agent negotiation to address the parking problem by taking into account not only motorists' preferences regarding parking locations, but also parking vendors preferences regarding car park occupancy, and social city benefits (e.g. less traffic congestion in city centers). Multi-agent negotiation was already used in Intelligent Transportation System applications, such as [1, 4]. In particular, in [1] cooperative agent negotiation is used to optimize traffic management relying on shared knowledge between drivers and network operators about routing preferences. In [4] agent negotiation is used for dynamic parking allocation, focusing on satisfying driver's preferences on prices and distances.

Here we use a flexible negotiation mechanism to find parking solutions that represent a compromise among different needs: a user who prefers to park close to the city center, the car park vendors who prefer to sell parking spaces in less occupied car parks, and a city manager who tries to limit the circulation of cars in city centers. At this purpose, a Car Park System is proposed in order to provide a coordinated selling of parking spaces belonging to different car parks, managed by a single software entity, the Parking Manager agent. We show that an automated negotiation mechanism between the Parking Manager and motorists represented by User Agents, allows to find a compromise solution for the involved negotiators, through the use of utility functions that model different needs that have to be dynamically evaluated, so helping users in their decision making process. The automated negotiation mechanism allows to formulate offers that do not strictly meet the user requirements, and to find parking solutions that are a result of a negotiation process between the PM and the UA upon parking attributes that are evaluated differently by the negotiators.

In principle, the proposed framework allows also to model different user's profiles since the evaluation of the parking space attribute values may vary for different classes of users. Furthermore, different UAs and different PMs may adopt different evaluation criteria respectively to reject/accept and to select offers that can be based on dynamic parameters, e.g. as the occupancy of the car park at the requested time, or the unavailability of public transportation at the requested time.

Finally, we showed that negotiation is a viable and promising approach since a solution that is found before all selected car parks are proposed to users, i.e. before they reach complete information on the parking spaces available offers, and that does not privilege only the drivers' preferences.

In order to better assess the usability of negotiation in real parking settings, a further experimentation is planned to evaluate the length of the negotiation process when the number of car parks increases and their occupancy distribution varies because of multiple users' requests. Also, more experimental settings have to be designed with different values of the UA threshold, modeling the user's "attitude" to reach an agreement, to evaluate their impact on the negotiation length.

Acknowledgements

The research leading to these results has received funding from the EU FP7-ICT-2012-8 under the MIDAS Project (Model and Inference Driven - Automated testing of Services architectures), Grant Agreement no. 318786, and the Italian Ministry of University and Research and EU under the PON OR.C.HE.S.T.R.A. project (ORganization of Cultural

	max_value	min_value	mean_value
# Available car parks	14	10	11 ± 2
# Rounds	9	1	3.3 ± 2.5
PM Utility	0.97	0.03	0.62 ± 0.22
UA Utility	0.75	0.10	0.68 ± 0.06
PM Utility 1 Round Neg			0.35 ± 0.27
UA Utility 1 Round Neg			0.71 ± 0.04

Table 2: Experimental Data collected in 150 runs.

HEritage for Smart Tourism and Real-time Accessibility).

6. REFERENCES

[1] J. L. Adler and V. J. Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5âĂŞ6):433 – 454, 2002.

[2] T. Baarslag, K. Hindriks, and C. Jonker. Acceptance conditions in automated negotiation. In T. Ito, M. Zhang, V. Robu, and T. Matsuo, editors, *Complex Automated Negotiations: Theories, Models, and Software Competitions*, volume 435 of *Studies in Computational Intelligence*, pages 95–111. Springer Berlin Heidelberg, 2013.

[3] M. Barbuceanu and W.-K. Lo. Multi-attribute utility theoretic negotiation for electronic commerce. In *Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems*, pages 15–30. Springer-Verlag, 2001.

[4] S.-Y. Chou, S.-W. Lin, and C.-C. Li. Dynamic parking negotiation and guidance using an agent-based platform. *Expert Syst. Appl.*, 35(3):805–817, Oct. 2008.

[5] C. Di Napoli, D. Di Nocera, and S. Rossi. Evaluating negotiation cost for qos-aware service composition. In *Proceedings of the 14th Workshop "From Objects to Agents" co-located with the 13th Conference of the Italian Association for Artificial Intelligence (AI*IA 2013)*, volume 1099 of *WOA '13*, pages 54–59. CEUR workshop proceedings, 2013.

[6] C. Di Napoli, P. Pisa, and S. Rossi. Towards a dynamic negotiation mechanism for qos-aware service markets. In *Trends in Practical Applications of Agents and Multiagent Systems*, volume 221 of *Advances in Intelligent Systems and Computing*, pages 9–16. Springer International Publishing, 2013.

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

[8] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *Int. Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[9] K. Nakamura, I. Hondo, N. Hataoka, and S. Horii. Car information systems for its. *Hitachi Review*, 49(3):102–106, 2000.

[10] B. Pan, J. Crotts, and B. Muller. Developing web-based tourist information tools using google map. In M. Sigala, L. Mich, and J. Murphy, editors, *Information and Communication Technologies in Tourism 2007*, pages 503–512. Springer Vienna, 2007.

[11] E. Polycarpou, L. Lambrinos, and E. Protopapadakis. Smart parking solutions for urban areas. In *2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, 2013.

[12] G. Svennerberg. *Beginning Google Maps API 3*. Apress, 2010.

[13] D. Teodorović and P. Lučić. Intelligent parking systems. *European Journal of Operational Research*, 175(3):1666–1681, 2006.