DeResolver: A Decentralized Negotiation and Conflict Resolution Framework for Smart City Services

Yukun Yuan, Meiyi Ma[□], Songyang Han[§], Desheng Zhang[†], Fei Miao[§], John Stankovic[□], and Shan Lin
Stony Brook University, [□] University of Virginia, [§] University of Connecticut, [†] Rutgers University

ABSTRACT

As various smart services are increasingly deployed in modern cities, many unexpected conflicts arise due to various physical world couplings. Existing solutions for conflict resolution often rely on centralized control to enforce predetermined and fixed priorities of different services, which is challenging due to the inconsistent and private objectives of the services. Also, the centralized solutions miss opportunities to more effectively resolve conflicts according to their spatiotemporal locality of the conflicts. To address this issue, we design a decentralized negotiation and conflict resolution framework named DeResolver, which allows services to resolve conflicts by communicating and negotiating with each other to reach a Pareto-optimal agreement autonomously and efficiently. Our design features a two-level semi-supervised learning-based algorithm to predict acceptable proposals and their rankings of each opponent through the negotiation. Our design is evaluated with a smart city case study of three services: intelligent traffic light control, pedestrian service, and environmental control. In this case study, a data-driven evaluation is conducted using a large data set consisting of the GPS locations of 246 surveillance cameras and an automatic traffic monitoring system with more than 3 million records per day to extract real-world vehicle routes. The evaluation results show that our solution achieves much more balanced results, i.e., only increasing the average waiting time of vehicles, the measurement metric of intelligent traffic light control service, by 6.8% while reducing the weighted sum of air pollutant emission, measured for environment control service, by 12.1%, and the pedestrian waiting time, the measurement metric of pedestrian service, by 33.1%, compared to priority-based solution.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Modeling and simulation**;

ICCPS '21, May 19–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8353-0/21/05…$15.00
https://doi.org/10.1145/3450267.3450538

KEYWORDS

Smart Services, Conflicts across services, Decentralized Resolution, Multiple Services Negotiation.

ACM Reference Format:
Yukun Yuan, Meiyi Ma[□], Songyang Han[§], Desheng Zhang[†], Fei Miao[§], John Stankovic[□], and Shan Lin . 2021. DeResolver: A Decentralized Negotiation and Conflict Resolution Framework for Smart City Services. In *ACM/IEEE 12th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2021) (ICCPS '21), May 19–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3450267.3450538

1 INTRODUCTION

The number of smart services has been increasing in modern cities. These services aim to improve the quality of urban lives, e.g., safety, wellbeing, and environmental quality. Examples of smart services include intelligent traffic light control [11, 39], air quality control [4], electric taxi scheduling [42], and ambulance management [13], etc. However, city managers are facing more and more potential conflicts across the growing number of deployed services [20, 23, 31]. For example, services may have different actions on the same devices due to self-interested objectives. Another example is that when acting alone, some city services are fine, but when combined they may be detrimental, e.g., to the environment. Such conflicts have significant impacts on the mobility and health of citizens.

Importantly, how to deal with potential conflicts across services is still under-explored. There exist several papers on resolving conflicts across smart services [18, 20, 24, 25, 35]. [35] uses a client-server architecture to choose one conflict resolution considering each application's specific performance requirements, e.g., resource consumption, and quality of services. [24] proposes a centralized conflict resolution for multiple city services by using an operation center to determine which actions are approved. [18] and [25] resolve conflicts in the smart home by assigning different priorities to smart applications based on their domains. However, these solutions have their intrinsic limitations: most of them require abundant and detailed knowledge of each service to determine the priority/weight, which is usually difficult to achieve in practice due to the private implementations of services; the rapid evolution of services makes keeping the decision center updated for all changes impractical; and it is hard for the center to understand and encode the complex operating logic of all services.

In this paper, we propose a novel decentralized negotiation and conflict resolution framework called DeResolver. Unlike the centralized conflict resolutions [18, 24, 25], DeResolver allows the services to resolve conflicts by communicating and negotiating with each other to automatically achieve a Pareto-optimal agreement. The decentralized design has several advantages. First, the decentralized

conflict resolution can ensure the privacy of services, i.e., without requiring the private information of services, e.g., objectives, service state, and actuator information, and it avoids the single point failure. Second, the decentralized design does not impractically require city managers to determine the importance or priority of an increasing number of services. Finally, most conflicts have the spatiotemporal locality. The spatial-temporal locality of conflicts means a conflict may only influence a local area of the city, and it may repeat multiple times during the upcoming time period after the first occurrence. Therefore, multiple conflicts may exist simultaneously but they usually influence different local areas of a city, or exist in different time periods of a day. It is natural and efficient to use a decentralized way to resolve each conflict independently.

The service negotiation problem provides a unique setting for an autonomous negotiation design. Services have access to local sensor data but do not know each other's objectives and utility functions. The services that compete against service i in a negotiation are called the opponents or opponent services of service i. To reach an agreement efficiently, it is essential for a service to learn about opponents' preferences under different situations. Therefore, a smart automated negotiation algorithm is designed to make accurate proposals based on the estimation of opponents' rankings of proposals. A semi-supervised learning algorithm is designed to predict acceptable proposals and their rankings for each opponent through historical negotiation records and corresponding states of the city. This design is evaluated with a case study of services from the domain of transportation and environment with real-world data-driven simulations using the Simulation of Urban MObility.

In summary, the **contributions** of this paper are as follows:

- To the best of our knowledge, we are the first to propose a decentralized negotiation framework, called DeResolver, for conflict resolution among city services. As service conflicts demonstrate high spatiotemporal locality in the physical world, the decentralized resolution allows smart services to mitigate cross-domain conflicts in an efficient and robust manner.
- We design a smart automated negotiation algorithm to perform automated negotiation and achieve a Pareto-optimal agreement. The automated negotiation algorithm is based on the estimation of how opponent services rank the configurations. We assumes that services do not know each other's internal state and utility function, which is different from the previous automated negotiation research, modeling utility functions of the opponents. We utilize the ranking of proposals reflected through historical negotiations to improve the accuracy and efficiency of negotiations.
- We design a two-level semi-supervised learning algorithm for estimating an opponent's rankings of configurations, the configuration ranking problem is different from state-of-the-art page ranking algorithms, as it is essential to classify proposals into acceptable and unacceptable sets under different states of the city besides providing a quantitative ranking estimation.
- Our data-driven evaluation is based on a dataset for vehicles that consists of the GPS locations of 246 surveillance cameras, and an automatic vehicle capture system with more than 3 million records per day. The results show that compared to a priority-based solution, our resolution can achieve a more equitable solution, i.e., only increasing the average waiting time of vehicles, the measurement metric of intelligent traffic light control service, by 6.8% while reducing the weighted sum of air pollutant emission, measured for environment control service, by 12.1%, and the pedestrian waiting time, the measurement metric of pedestrian service, by 33.1%.

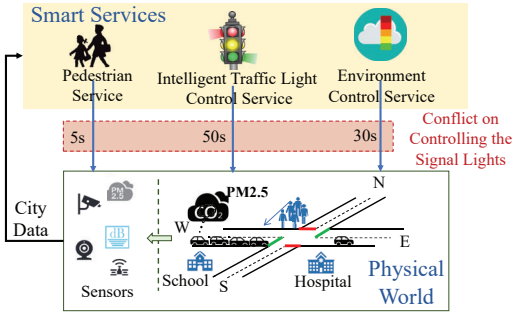


Figure 1: Demonstration of Example

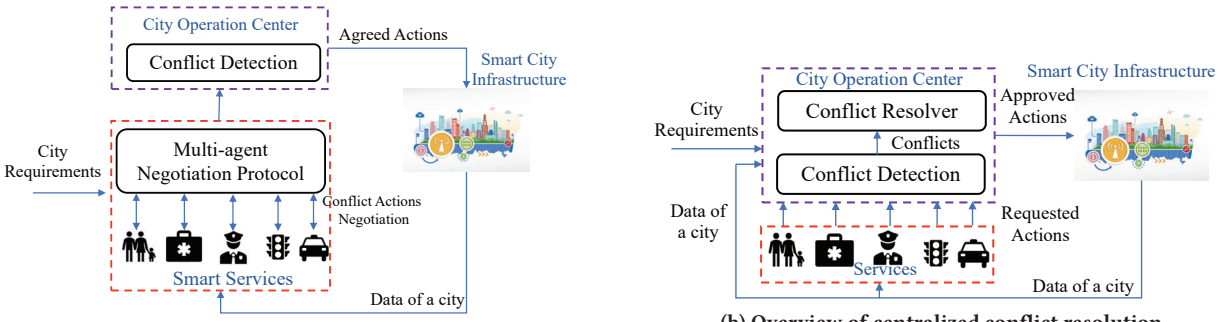
2 CONFLICTS ACROSS CITY SERVICES

2.1 Motivating Example

Modern cities have already implemented smart services to enhance the quality of citizens' lives. These services may be provided by the different companies or departments to the city government. For example, the city bike company dispatches bikes around the city to provide the last-mile transit service [36], the taxi company provides the ride-sharing service [43], and the public safety department schedules patrols to defend against potential attackers [41]. However, conflicts across services arise when two services cannot perform actions simultaneously due to undesirable and harmful effects. In this work, we introduce and use the following example to better illustrate the definition and real-world application scenario of conflicts across services, the scope of the problem that this work addresses, and the system design.

Example 2.1. It shows the inconsistent configurations on traffic lights by three decentralized services. Intelligent traffic light control service [11, 38]: it configures the traffic lights to minimize the average traffic delay at the road intersections. This service can be a decentralized service [38] configuring a traffic light for a intersection independently due to the high computation complexity of coordinating multiple traffic lights simultaneously. It is in the transportation domain. Pedestrian service [29]: it sets up the traffic lights that show pedestrian crossing signals to minimize the average pedestrian waiting time. This service implements a controller to configure the traffic lights for pedestrians at a road intersection independently. The reason is the setup of a traffic light for pedestrians has little influence on the setup of another one at a nearby road intersection due to the limited walking distance of pedestrians. It is in the transportation domain. Environment control service [4]: it controls the traffic lights to raise environmental quality, e.g., increasing air quality and decreasing noise levels, of road segments. It is a decentralized service and in the environment domain.

These three services run concurrently to achieve their respective objectives. However, potential conflicts may exist among them at run-time. Three services determine their configurations of the green light interval of the West-East (W-E) or North-South (N-S) directions. The configurations of traffic lights for pedestrians and vehicles should be consistent. The intelligent traffic light control



(a) Overview of decentralized conflict resolution

(b) Overview of centralized conflict resolution

Figure 2: Comparison of centralized and decentralized conflict resolution

wants a long green light duration due to the high waiting traffic from W-E, whereas the pedestrian service wants to configure a short green light interval because of a few pedestrians in the N-S direction. The environment control service does not desire a long green light duration because the accumulated vehicles around a hospital would increase the air pollution, nor desire a short interval due to the increment of noise level from congested vehicles near a school. Therefore, to meet individual service performance requirements, the conflicts exist among these services.

Example 2.2. Another example is three services desire different amount of traffic on certain road segments and cause conflicts. Event service: it blocks the lanes nearby the event to reduce traffic near a city event. Parking service: it navigates the drivers to parking lots near the event. Detour service: it navigates traffic around a road segment under emergency repairs near the event. A conflict across the three service occur when these three services decide how much traffic can be directed to a certain road segment. Coordinated actions are needed to minimize the local congestion.

Based on the examples, we define the conflicts across services as *if two or more services have inconsistent actions on the shared resources due to incompatible individual goals, they have a conflict.* If conflicts are not resolved and managed equitably, they can affects citizens' daily lives. We note that such conflicts do not happen very often for well designed smart services, they usually occur when 1) new services and requirements are deployed, 2) city environment changes, and 3) disruptive and unexpected events happen.

2.2 DeResolver Framework

In this work, we consider the following setting of services that result in the conflicts across services. Each service is provided by a stakeholder and has the self-interested and private control objective, which is usually not completely known by the other services. For instance, any service in Example 2.1 do not know the exact control models of the other services. The services can access data from the deployed sensors to check the specific state of the city, e.g., NYC [27] and Newark [28]. Services can reliably communicate and share information with each other since services managed by different stakeholders have already communicated with the city center to report the operational data in the existing city systems.

To resolve the conflicts across services under the above setting, we design a decentralized negotiation based conflict resolution, DeResolver. Figure 2a shows an overview of DeResolver and it works in three steps. First, smart services collect the data of the city using deployed sensing devices to determine the control decisions and send them to the city operation center. After receiving the control decisions, the operation center uses a conflict detector, e.g., CityGuard [22] to check whether a conflict exists. If a conflict is detected, the center notifies all the involved services that their collective control decisions result in a conflict and they should resolve the conflict by DeResolver; otherwise, the operation center applies the received control decisions.

Second, with DeResolver, the *services that result in the conflicts* are organized to negotiate an *agreement on the configuration of the shared resources* based on a carefully designed multi-agent negotiation protocol. In each negotiation period, a randomly selected service makes its proposal of the action (i.e., new control decisions) to the other services. Please refer to Section 4.2 for how to make the proposal. Then each of the other services answers acceptance or rejection for the proposal and their answers are broadcasted to all the services in the negotiation. Please refer to Section 4.3 for how to make the acceptance or rejection decision. If a proposal is agreed upon by all the services, they reach an agreement and the negotiation terminates; otherwise, the negotiation continues until the deadline is reached. Section 3 introduces how to define the deadline of the negotiation based on the application scenario.

Finally, when an agreement is achieved, it is sent to the operation center, which will detect whether the agreement results in a new conflict considering potential new control decisions from other services that were not in the previous negotiation process. If not, the operation center applies the agreement; otherwise, the center notifies all the services resulting in the new conflict that they need to repeat the second step to resolve the new conflict. If the services do not reach an agreement, the city operation center will execute the default action on the shared resources, which is unknown to the services that result in the conflict.

Existing work [24] proposed CityResolver, a centralized resolution for conflicts across services. Figure 2b shows the overview of it. Services send their requested actions to the city operation center. Then the center detects whether conflicts exist using CityGuard [22]. If so, the conflict resolver approves part of requested actions to generate a group of actions without conflicts based on its objectives, and then apply the approved actions. If no conflict is detected, all requested actions are applied to the actuators. There are two features of CityResolver. The first one is that it simultaneously addresses all the conflicts using an integer linear programming based method if these conflicts happen at the same time. The second feature is that it assigns a weight to each requested action that is determined by current state-dependent importance policies.

We summarize the reasons of using the decentralized negotiation based solution to address the conflicts across services as follows. (i) The decentralized negotiation based resolution can avoid single

point of failure by relying on services rather than a central agent to resolve conflicts. Meanwhile, our decentralized solution does not require the services to upload any private information, ensuring the privacy of services and actuators. (ii) The decentralized design does not impractically require city managers to determine the importance or priority of an increasing number of services.

(iii) A centralized solution may not be efficient to resolve multiple conflicts happening at the same time in a city. It may not perform well due to the curse of dimensionality in the joint action space of actuators [38], e.g., using a single agent to control hundreds of traffic lights. In general, it is not necessary to consider many simultaneous conflicts together in a centralized optimization, which is computationally challenging and resource demanding. Because these conflicts may happen in the different local areas of a city and they do not affect each other.

3 DERESOLVER FRAMEWORK DESIGN

In this section, we demonstrate the formulation of the negotiation for addressing the conflict in Example 2.1. Please see Appendix A.1 for the formulation of DeResolver on addressing a general conflict.

The traffic lights for pedestrian crossing signals or vehicle traffic coexist in a road intersection. The signals provided by these two types of traffic lights should be consistent to avoid traffic accidents. Therefore, we assume that there is a traffic light at the intersection of two roads. To simplify the notation, north, south, west, and east are represented by "N", "S", "W", and "E" respectively, and "Green" and "Red" are used to describe the green and red light. Since two roads' traffic cannot pass the intersection at the same time, there are two states of a traffic light, i.e., (1) Green-WE (Red-NS) and (2) Red-WE (Green-NS).

In a real-world scenario, such two states exist alternatively, i.e., $1 \rightarrow 2 \rightarrow 1 \rightarrow \ldots$, meaning the schedule of a traffic light is a sequence of phases, where a phase represents several consecutive time slots when a traffic light has the same state. In the first motivating example, each service sets up the length of each traffic light phase, e.g., the number of seconds of each traffic light phase. To simplify the problem description, let t be the traffic light phase that a traffic light is within, and services negotiate the configurations of $(t+1)$-th traffic light phase.

Negotiation agent: A negotiation is organized for resolving a conflict, and it consists of N services whose control decisions result in a conflict. A negotiation agent represents a service. For Example 2.1, the negotiation is played by three services ($N = 3$). These N services negotiate the issue under discussion within H periods.

State: The three services access data from the deployed sensors to check the specific state of the city that they are interested in. Figure 1 shows the specific sensors that three services use to collect the data of a city. We list the information that the different sensors can provide as follows: road surveillance camera: videos of traffic around the road intersections; vehicle loop detector: vehicles count; air quality sensor: air quality value; noise sensor: noise level; pedestrian crossing surveillance camera: videos of pedestrians close to the pedestrian crossing. Let $s_{i,k}(t)$ be the states of the city around the traffic light k at the beginning of phase t that service i is interested in, and we assume the above states around a road intersection are stable during a negotiation. Due to the space limitation, please see Appendix A.2 for the state of the city that three services take.

Proposal: The issue under discussion is defined as the configuration of a traffic light. The proposal is the length of the next traffic light phase. During a phase t, three services negotiate the traffic light configurations for $(t+1)$-th phase. Service i proposes $O^h_{i,k}(t) \in D$ during the negotiation period h to configure the phase t of traffic light k. The domain of a traffic light's phase length is defined as $D = \{d |\ d \in [T_{min}, T_{max}] \text{ and } d \in \mathbb{Z}_+\}$, where T_{min} and T_{max} correspond to the extreme values of the phase duration.

Agreement: In this example, the agreement means there exists a proposal, $O^h_{i,k}(t)$, of traffic light k's configuration, which is approved by all other services during the negotiation period h.

Utility: Let $r_i(O^h_{i',k}(t))$ be the immediate utility that service i can get if the proposal $O^h_{i',k}(t)$ is applied to the t-th traffic light phase. The utility functions of three services are formulated as follows. Please see Appendix A.3 for the definition of utility functions for three service. In brief, intelligent traffic light control service aims to minimize the waiting time of traffic, pedestrian service would like to reduce the pedestrian waiting time, and environment control service focuses on optimizing the environment quality, e.g., reducing the noise level and air pollutant emission.

Multi-agent negotiation protocol: If these three services have multiple simultaneous conflicts on the configurations of n traffic lights, then they play n negotiations simultaneously, where a negotiation is organized for resolving a conflict on one traffic light's configurations. Finally, an agreement is achieved for each negotiation, and multiple simultaneous configurations are agreed among these services. During a period h of the negotiation for traffic light k, a service proposes its configuration to its opponent services. Then the other services determine to accept or reject this proposal based on their own interests. If this proposal is accepted by the other services, it is an agreement and will be applied to the actuators; otherwise, the negotiation moves to period $h+1$, and another service proposes its configuration again.

Services make the proposals by the round-robin principle during different negotiation periods. It means that only a service proposes its solution to the issue under discussion during a negotiation period and N services make the proposals in a circular order. If a service makes a proposal during period h, it should make another proposal during the negotiation period $h+N$ as long as neither an agreement is reached nor the negotiation terminates. If there exists a negotiation agent that rejects the proposal O^h_i, the negotiation moves to the period $h+1$, and another service makes its proposal. The negotiation process terminates when an agreement is reached or the number of negotiation periods is over H. We use an example to demonstrate how to define H in traffic light control. For instance, the configuration of the green light phase duration of the N-S direction should be determined before the green traffic light phase of E-W direction ends. Then according to the starting time of the negotiation and the deadline, the maximum duration of the negotiation is obtained. H is equal to the maximum duration of the negotiation over the length of a time period, where the denominator can be a static value, e.g., a half seconds. If no agreement is achieved, the default configurations is applied to traffic light k, which are determined by city transportation authorities.

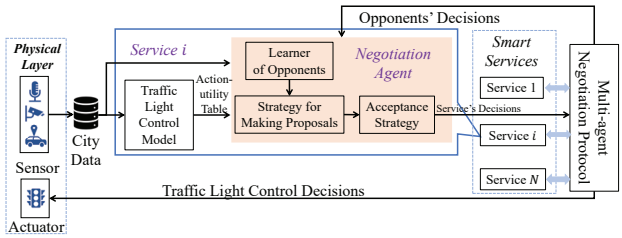


Figure 3: Design of a service under DeResolver Framework

4 DESIGN OF A SMART SERVICE UNDER DERESOLVER FRAMEWORK

It is essential for a service to optimize its negotiation strategy to maximize its utility decided by the agreement. To address this problem, we design an automated negotiation agent for each service that determines the negotiation actions by learning its opponent services' acceptable configurations and ranking for different configurations from the past negotiation behaviors.

Definition 4.1 (Automated Negotiation Problem). Given services with conflicts and the negotiation protocol formulated in Section 3, the problem is how any service i determines its action at any negotiation period h, i.e., accepting or rejecting the proposal from other services, and making its proposal, to maximize its utility.

Figure 3 shows the design of a service with an automated negotiation agent under the DeResolver framework. We take the traffic light control as an example. Given the data of a city, the traffic light control model is used to estimate the action-utility table based on the utility function of service i. There are two columns of each row in the action-utility table, where the first column represents a possible traffic light configuration (action), and the second column is the long-run utility that service i receives if applying the action to the traffic light. Then service i determines its action at each negotiation period h according to the estimation that which configurations are acceptable to service i' and how an opponent service i' ranks the different estimated acceptable configurations. In detail, the learner of opponents outputs a sequence of acceptable configurations to each opponent service i', and this sequence has the ranking information. The proposal strategy determines the proposals that service i make, and the acceptance strategy decides to accept or reject opponent services' proposals.

4.1 Learner of Opponents

Service i should have some beliefs of its opponent services through the negotiation to maximize its utility that negotiation result introduces. We propose a two-levels learner for service i to estimate how any opponent service i' ranks the acceptable configurations. In detail, service i needs to learn $N-1$ models, where a model corresponds to an opponent service i'. The first level of the learner estimates a set of acceptable configurations to service i' based on the current state of the city and negotiation behaviors during the previous periods of the current negotiation. The second level infers how service i' ranks the estimated acceptable configurations.

4.1.1 Estimating acceptable configurations. The task in the first level of the learner is to learn a function $f^1_{i'}$(state during a period, configuration) $\in [0,1]$ showing the probability that opponent service i' accepts configuration $d \in D$ during period h given the state during period h. The first-level of the learner takes the state during a negotiation period and a configuration as the input. This function works as a binary classifier to estimate whether a configuration is accepted or rejected by service i'.

First, we define the state during a negotiation period h, denoted as $ns^{1,h}_{i'}$. $P^h_{i'} = \{0,1\}^{1\times|D|}$ represents whether the configurations are accepted or not by service i' before period h. If configuration d_l is proposed or accepted by service i' before period h, $P^h_{i',l} = 1$; otherwise, it is 0. We define $ns^{1,h}_{i'} = (s_{i'}(t), P^h_{i'})$, which is a concatenation of the state of a city that service i' takes and the indicator matrix showing whether configurations are accepted or not.

Second, we discuss how to generate the labeled training data denoted by $U^1_{i'}$ with a form of <state, configuration, label> for each data sample to learn the binary classifier. Given the records of a past negotiation, we generate $ns^{1,h}_{i'}$ based on its definition during each period h. Given $ns^{1,h}_{i'}$, if a configuration d is accepted, i.e., the label y is 1, we have a data sample, i.e., $< ns^{1,h}_{i'}, d, 1 >$; otherwise, the data sample is $< ns^{1,h}_{i'}, d, 0 >$, i.e., the label y is 0.

We train $f^1_{i'}$(state during a period, configuration) using the labeled data to minimize the following loss function:

$$\mathcal{L}^1 = -\frac{1}{|U^1_{i'}|} \sum_{<ns^{1,h}_{i'}, d, y>} y \log(f^1_{i'}(ns^{1,h}_{i'}, d)) + (1-y)\log(1 - f^1_{i'}(ns^{1,h}_{i'}, d)) \quad (1)$$

This cross-entropy loss function is widely used for binary classification problems. This function calculates a score that summarizes the average difference between the actual and predicted probability distributions. If the actual classification value is 0, the corresponding loss value is $-\log(1 - f^1_{i'}(ns^{1,h}_{i'}, d))$; otherwise, the loss value is $-\log(f^1_{i'}(ns^{1,h}_{i'}, d))$. The optimal cross-entropy loss value is 0. There are multiple binary classifiers widely used in the related work, e.g., neural network, K-nearest neighbors and support vector machines. In the evaluation, we set their loss functions as Equation (1), and then evaluate their performance. The classifier that generates the best results empirically is used in the data-driven evaluation. The set of acceptable configurations to an opponent service i' may change under the different state of the city, e.g., dynamic traffic volume in each direction, and our training function adapts to such changes since the state of a city is a part of its input.

4.1.2 Estimating ranking of acceptable configurations. Given the set of acceptable configurations to service i', service i still needs to estimate how service i' ranks these estimated acceptable configurations. An intuition is that the opponent service i' ranks the configurations based on the utility that they introduce. The task of the second level of the learner is to learn a function $f^2_{i'}$(state, configuration) that ranks the estimated acceptable configurations. We assume that the ranking function assigns a score $f^2_{i'}(\cdot)$ to each configuration, where a large score represents a high ranking. The inputs to this learner are the state of the city that service i' is interested in and a configuration.

To learning the above score function, we first generate our training data including data with ranking information $U^2_{i'}$ and data without ranking information $L^2_{i'}$ of service i'. During any past negotiation, service i' may propose multiple configurations. We assume that for any two configurations, service i' proposes the one with higher utility at first. Suppose a past negotiation is associated with a stable state of the city, denoted as $ns^2_{i'}$. In a negotiation, if any two configurations d_1 and d_2 are proposed by service i' during two different periods and d_1 is proposed earlier, we add a data sample,

i.e., $< l_1 > l_2 >$ where $l_1 =< ns^2_{i'}, d_1 >$ and $l_2 =< ns^2_{i'}, d_2 >$ to $U^2_{i'}$. In the same negotiation, if any two configurations d_1 and d_2 are not proposed by service i', we add a data sample $< l_1, l_2 >$ to $L^2_{i'}$. The ranking information is included in any sample in $U^2_{i'}$. However, it is not contained in $L^2_{i'}$.

We use a semi-supervised learning method to train the function $f^2_{i'}(\cdot)$ by these two datasets. For the training dataset $U^2_{i'}$, we consider the probability models that assign a probability of $< l_1 > l_2 >$, based on the score difference $f^2_{i'}(l_2) - f^2_{i'}(l_1)$.

Bradley-Terry model [17, 34] is widely used to estimate the probability $P(< l_1 > l_2 >)$ that $< l_1 > l_2 >$ is true given a pair of individuals l_1 and l_2. This model associates a score $f^2_{i'}(l_1)$ with each individual configuration, l_1. Then it defines the probability that l_1 is preferred to l_2 as the logistic function of their score difference:

$$P(< l_1 > l_2 >) = \frac{1}{1 + e^{f^2_{i'}(l_2) - f^2_{i'}(l_1)}}$$

Therefore, the objective of training dataset $U^2_{i'}$ is to maximize the following likelihood function:

$$\sum_{<l_1>l_2>\in U^2_{i'}} \log(P(< l_1 > l_2 >)) = \sum_{<l_1>l_2>\in U^2_{i'}} \log \frac{1}{1 + e^{f^2_{i'}(l_2) - f^2_{i'}(l_1)}} \quad (2)$$

Since ranking information is not included in $L^2_{i'}$, we would like to tie the similarity of configurations to the score similarity. Let r_{l_1,l_2} represent the similarity between l_1 and l_2, defined as $r_{l_1,l_2} = \frac{|d_1 - d_2|}{T_{max} - T_{min}}$. Then we would like to penalize the function $f^2_{i'}(\cdot)$ if similar configurations have quite different scores, formulated as:

$$\sum_{<l_1,l_2>\in L^2_{i'}} -r_{l_1,l_2} \log\Big(P(< l_1 \not> l_2 >)P(< l_2 \not> l_1 >)\Big) = \sum_{<l_1,l_2>\in L^2_{i'}} -r_{l_1,l_2}$$

$$* \log\Big((1 - P(< l_1 > l_2 >)) * (1 - P(< l_2 > l_1 >))\Big) = \sum_{<l_1,l_2>\in L^2_{i'}} -l_1, l_2 \log \frac{0.5}{1+cosh(f^2_{i'}(l_1) - f^2_{i'}(l_2))}$$

The intuition of the above equation is that for two configurations without preference information, the probability function of preference should not show that $< l_1 \not> l_2 >$ or $< l_2 \not> l_1 >$. The first part of the above equation ensures that if there is no preference between l_1 and l_2, the penalty is minimized when $P(< l_1 > l_2 >) = 0.5$. If $P(< l_1 > l_2 >)$ is close to 0 or 1, the penalty is maximized. Then we apply the Bradley-Terry model to generate the right side.

In summary, we would like to train the function $f^2_{i'}(\cdot)$ to maximize following function with a negative weight β to balance the importance of fitting two datasets:

$$\mathcal{L}^2 = \sum_{<l_1>l_2>\in U^2_{i'}} \log \frac{1}{1+e^{f^2_{i'}(l_2) - f^2_{i'}(l_1)}} + \beta \sum_{<l_1,l_2>\in L^2_{i'}} -r_{l_1,l_2} \log \frac{0.5}{1+cosh(f^2_{i'}(l_1) - f^2_{i'}(l_2))} \quad (3)$$

The trained function $f^2_{i'}(\cdot)$ assigns a score to these estimated acceptable configurations, and then service i estimates how service i' ranks them. We do not assume that the ranking of configurations is stable for service i' since the ranking may change with the state of a city, e.g., dynamic number of waiting vehicles or pedestrians in each direction. Our learning functions take the dynamic state of a city as a part of its input, so they can estimate the new ranking of configurations for service i' when the state of a city changes.

4.2 Strategy for Making Proposals

In this part, we introduce our strategy for making proposals designed for service i. This strategy takes the action-utility table, past negotiation behaviors of current negotiation, and the estimation that how opponent services rank the estimated acceptable configurations as input to determine the proposal.

A service wants to achieve an agreement to get as much utility as possible by using a strategy to propose a configuration which not only introduces the highest utility to itself but also is acceptable to the other services based on the estimation of opponents. If such a configuration does not exist, the service lowers its lowest acceptable utility to make a proposal, and the amount of utility that is given up depends on its opponent services' last two proposals.

When the learner of opponents is used by service i to estimate how service i' ranks the estimated acceptable configurations during period h, let $A^h_{i,i'}$ denote the output of learner and it is a sequence of acceptable configurations.

Since service i's last proposal is rejected by at least one of other $N-1$ services, service i should concede its lowest acceptable utility to make its proposal be acceptable to other services. We use the reactive concession strategy for service i to update its lowest acceptable configuration based on the previous proposals of other $N-1$ services. Service i computes the ranking difference, which represents how much any opponent service i' concedes between its last two proposals, denoted as $\Delta u^h_{i,i'}$ based on the estimation of how service i' ranks the configurations. The maximum ranking difference that service i can decrease is equal to $\min_{1\le i'\le N, i'\neq i} \Delta u^h_{i,i'}$. Thus, based on service i's last proposal, maximum ranking decrease, and its action-utility table, service i updates its list of the acceptable configurations during period h, denoted as $A^h_{i,i}$. It is noted that service i only concedes during the negotiation periods when this service makes the proposal. We assume that $A^h_{i,i}$ only includes the configuration with the highest utility when service i makes the first proposal of current negotiation.

Proposal generation: Since the negotiation may terminate with different agreements, we use the Pareto-optimal agreement to measure them and the definition is shown as follows:

Definition 4.2 (Pareto-optimal agreement). One agreement d is Pareto-optimal if there is no other agreement d' such that for utility function U_i for agent i, $\forall i \in \{1, ..., N\}, U_i(d') \ge U_i(d)$ and $\exists\, i, U_i(d') > U_i(d)$.

In other words, a Pareto-optimal agreement is able to make any individual service's performance better off without making at least one individual service's performance worse off. There may be multiple Pareto-optimal agreements of a negotiation, and we do not measure which one is the best. Service i also wants to reach a Pareto-optimal agreement since such a result can maximize its performance, and it does not make opponents miss any benefit.

During period h, if service i makes its first or second proposal, it can choose the configuration with the highest or $(S+1)$-th highest utility respectively, where S is the initial concession rate. Otherwise, it first lowers its lowest acceptable utility using the reactive concession strategy to get the new list of acceptable configurations, $A^h_{i,i}$. Let $\mathcal{I}^h_i$ be the set of configurations that exist in all configuration lists $A^h_{i,i'}$ $(1 \le i' \le N)$. If the set $\mathcal{I}^h_i$ is not empty, service i s the following configuration during period h: $d = \text{argmax}_{d\in\mathcal{I}^h_i \cap Pe} U_i(d)$, where Pe is the set of Pareto-optimal agreements calculated by service i based on its estimation of other services' ranking of configurations. Although service i does not know the actual utility functions of opponent services, the estimated ranking of configurations is enough to compute the set of Pareto-optimal solutions. If

I_i^h is empty, meaning there is no configurations that are acceptable to all services, service i proposes the configuration introducing the lowest acceptable utility since it has already conceded when updating the set of acceptable configurations, and the proposed configuration is defined as $\text{argmin}_{d \text{ in } A_{i,i}^h} U_i(d)$,

4.3 Acceptance Strategy

In this part, we describe the strategy that service i uses to determine acceptance or rejection of a proposal from an opponent service i' during period h. Our negotiation agent uses a utility-based condition to make the decisions. Given the proposal from another service i' during period h denoted by $O_{i',k}^h$, service i first checks whether $O_{i',k}^h$ is an element of $A_{i,i'}^h$. If not, service i rejects this . Otherwise, this service detects whether $O_{i',k}^h$ can be improved, meaning there is another configuration d which can improve the utility of service i and does not decrease the performance of service i' according to $A_{i,i'}^h$ and the action-utility table of service i. If $O_{i'}^h$ can be improved, service i should reject it since its utility can be improved while not sacrificing other services' performance. Otherwise, it is accepted.

4.4 Negotiation Agent Design

According to the strategy for making proposals and the acceptance strategy, we summarize the automated negotiation agent as follows: if it is service i's turn to make a proposal during period h, service i will use strategy for making proposals to determine its proposal; otherwise, service i decides to accept or reject a proposal from another service using the acceptance strategy.

Theorem 4.1. *Consider N services in a negotiation, they negotiate to determine a configuration from the configuration space $D = \{T_{min}, T_{min}+1, \cdots, T_{max}\}$. If all N services use our negotiation algorithm, and the estimation of acceptable configurations and their ranking to any service i is accurate, the result of our algorithm is guaranteed to reach a Pareto-optimal agreement.*

Proof. Please refer to Appendix A.4. □

4.5 Action-utility Table Computation

It is essential for any service i to know the utility that different configurations can introduce. Even without conflicts, each service also needs to compute such a table to choose the optimal configuration for optimizing the quality of service. The action-utility table computation is beyond the scope of this study. Please refer to Appendix A.5 for how we obtain the table for services in Example 2.1.

5 VALIDATION

5.1 Methodology

The experiments are conducted using SUMO, a simulation platform providing APIs to model traffic systems including vehicles, pedestrians, environment measurement, and traffic light control. Specifically, SUMO can simulate vehicles and pedestrian mobility for given routes and traffic light control policies.

We collect the real-world vehicle mobility data by 246 surveillance cameras in Shenzhen, China over the time period from 05/01/2017 to 05/20/2017. A record is generated when a vehicle is captured by the camera, and each record consists of captured time, camera ID, and other information. We also have the city map to show each road intersection's GPS data and a table to map each camera ID

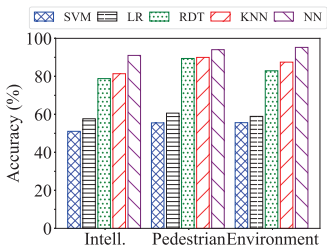


Figure 4: Performance of classifying acceptance or rejection

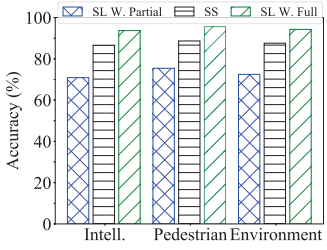


Figure 5: Performance of learning opponents' preference

to the actual GPS location. The size of the dataset is 55.0 GB. We use five-days data for the experiment and the remaining data is imported into SUMO for training. We import a $3\ km \times 2\ km$ region of Shenzhen and the corresponding vehicle traffic as the city environment to SUMO, including nine traffic lights that three services want to control. We also generate pedestrian traffic with the setting that one person per four seconds or five seconds for the different directions of a intersection. A traffic light is configured as an integer between 1 and 60, representing the duration of a traffic light phase.

To evaluate the performance of DeResolver, we compare it with four state-of-the-arts centralized conflict resolution frameworks: (i) a priority-based solution based on [25], it gives a higher priority to intelligent traffic light control service than the other two services, because the primary goal of traffic light control is traffic flow optimization. (ii) A weight-based solution based on [24], it selects the action that maximizes the weighted sum of three services' utility ratios. The utility ratio of a service is defined as the ratio between the utility of an action and the maximum utility that this service can get if no conflict occurs. The weight is determined by city managers according to their understandings of services. The weights for intelligent, pedestrian and environmental control services are defined as 1, 2, and 10 respectively. (iii) A round-robin solution that applies the requested actions from three services by cyclic execution. (iv) A Pareto-efficient solution that knows the action-utility tables of all services and selects the Pareto-efficient configuration that maximizes the minimum service utility ratio of all services. This solution ensures that any service cannot improve its performance without reducing any other services' performance. Meanwhile, it provides the max-min fairness for services.

Taking the Example 2.1, we define the metrics to measure three services' performance. Average waiting time of a vehicle: for a vehicle, we calculate its waiting time (speed less than 0.1 m/s) and report the average value. Average waiting time of each pedestrian: we calculate the waiting time of each pedestrian and report the average value. The weighted sum of air pollutant emission per hour: we assign a weight for different road segments based on the nearby environment, e.g., large weight for hospitals and schools, and then report the weighted sum of air pollutant emission of all road segments. The measurement unit of waiting time is second, and that of environment control service is kilogram per hour.

5.2 Performance of learner of opponents

First, we describe how we collect the data used to train the learners that are proposed in Section 4.1. We simulate that three services operate to control nine traffic lights by feeding the fifteen-day traffic

data into SUMO, and we set up that they play a negotiation if conflicts exist. During the negotiation, each service uses the proposed negotiation agent to play the negotiation, and services also collect the data that is generated from services' negotiation behaviors. In each period of a negotiation, a data sample is generated for each service following the process described in the third paragraph of Section 4.1.1. The data samples collected over all periods of all negotiations, are used to train the binary classifier. Two datasets, i.e., one with ranking information and the other one without ranking information, are generated when a negotiation ends with the process described in the second paragraph of Section 4.1.2. Two datasets are used to train the learner that is for inferring the ranking.

Second, we define the estimation accuracy as the main metrics for evaluating the learning based algorithm. The accuracy of estimating the acceptable or unacceptable configurations to service i is used to measure the performance of the first level of the learner. The accuracy of estimating how service i ranks the acceptable configurations is used to measure the performance of the second level of the learner. Due to the space limitation, please refer to Appendix A.6 for the detailed mathematical description of the estimation accuacy.

Finally, we report the evaluation results. We measure the performance of five widely used binary classifiers, and then select the one with the best performance. Support vector machine (SVM): a classical algorithm to find a hyperplane for classifying the data. Logistic regression (LR) [14]: a statistical model estimating the parameters of a logistic model. Random decision tree (RDT) [9]: a method that constructs multiple trees in randomly selected subspaces of the feature space and uses the combined predictions of the individual trees as the output. K-nearest neighbors (KNN) [1]: a type of instance-based learning, where the classification of a data point is the same as the class most common among its K nearest neighbors. In this evaluation, we empirically test the value of K, and set K as 10 because $K = 10$ has the best results in our tests. Neural network (NN) [7]: it uses a NN to learn the linear or non-linear combination.

Figure 4 shows the classification accuracy using five different classifiers to estimate whether any one of three services accepts a configuration or not. It can be observed that the neural network based learner outperforms all other four solutions with more than 90.0% accuracy for all three services, which means that more than 90.0% of configurations are correctly classified as acceptable or unacceptable to an opponent service. The reason is that a neural network can approximate both linear and non-linear hyperplane to partition the feature space. It is also observed that KNN also achieves the second best performance with accuracy more than 80.0% since any two close configurations have a high possibility to receive the same acceptance or rejection decisions. In conclusion, we use a neural network based classifier to estimate whether configurations can be accepted by a service.

When measuring the semi-supervised learning algorithm (SS), we design two other methods for comparison: supervised learning with partial order information (SL W. Partial) and supervised learning with full order information (SL W. Full). The first method learns the opponent ranking model only using the collected data with partial order information and aiming at only minimizing logistic loss function, i.e., Equation (2). The second comparison method assumes that given the state of the city, the full order information of all configurations is known, and this method conducts supervised learning to minimize Equation (2). We use TF-Ranking [30] to implement our semi-supervised learning method, which optimizes the weighted sum of two objectives simultaneously.

Table 1: Performances of different conflicts resolutions. (Heavy traffic means the rush hours of one day and light traffic means non-rush hours of one day.)

	Intell.		Environment		Pedestrian	
	Light	Heavy	Light	Heavy	Light	Heavy
Priority-based	161.48	300.56	22.93	33.75	565.20	3085.20
Weight-based	293.74	530.86	18.40	26.17	309.60	1613.88
Round-robin	289.74	518.65	20.05	30.62	334.08	2177.64
DeResolver	172.39	412.59	20.15	31.11	378.00	1890.31
Pareto-efficient	172.61	381.96	20.18	29.01	345.20	1701.72

Figure 5 shows the estimation accuracy of order between any two configurations by three methods. It is observed that our semi-supervised learning method can achieve more than 86.0% accuracy for estimating the preference of all three services. For pedestrian service, our semi-supervised learning algorithm increases the estimation accuracy by 17.6% compared with the supervised learning method with only partial order information. But it decreases the performance by 7.2% compared with the supervised learning algorithm with full order information. This observation is normal since our solution takes full use of the distance between two configurations without order information to penalize generating a large score difference for two close configurations. However, a dataset with full order provides the most information.

5.3 Performance of DeResolver

5.3.1 Comparison of resolutions to eliminate conflicts. Table 1 shows the performance of three services when using different solutions to resolve conflicts across them under the light and heavy traffic. We set up that all three services using our negotiation method under the DeResolver framework. There are two observations. The first one is priority or weight based resolution can improve the performance of one service greatly, meanwhile degrading the other two services' performance significantly. By comparing priority and round-robin solutions, we can see that changing the subjective weight on different services could reduce the weighted sum of air pollutant emission by 40.9% while increasing the average vehicle waiting time by 79.4% with light traffic. The second one is our solution can achieve close performance compared with the Pareto-efficient solution. Compared with applying a Pareto-efficient solution which is generated from the actual action-utility table of three services, DeResolver can achieve close vehicles and pedestrian waiting time, meanwhile increasing the air pollutant emission by 9.5% with light traffic. Meanwhile, with heavy traffic, the performance of three services by DeResolver decreases less than 10% compared with that of three services by Pareto-efficient solution. It is because DeResolver misses the action which can maintain the performance of two services and improve that of environment control service due to estimation error.

The action conflicts across services result in a trade-off among these services' performance when determining the configurations of shared actuators. To be noted, DeResolver's goal is to find a trade-off for all the services' performance. In some cases, DeResolver cannot guarantee the best performance of each individual service compared with some other algorithms. However, it does provide a balanced performance while resolving the conflicts (as shown in Table 1). Different from the solutions that assign a higher

priority or weight to a particular service based on prefixed rules, DeResolver allows smart services to reach agreements under dynamic city states via negotiation, introducing a balanced set of actions for all the services. We believe this is an important step to maintain balanced performances of smart services, especially with an increasing number of services deployed in smart cities. In the future work, we will continue exploring the fairness when resolving conflicts among services.

5.3.2 Comparison of DeResolver's variations. We evaluate whether learning services' ranking is useful for improving the performance of one service. We consider two variations The first one is DeResolver w. perfect opponent learning. Suppose one service knows the acceptable configurations to opponent services and the corresponding ranking. The second variation is DeResolver w/o opponent learning. Suppose one service does not learn opponent services' acceptable configurations and preference of different actions. It just proposes the configurations from the one with the highest utility to the one with the lowest utility one by one. In this experiment, we assume that the intelligent traffic light control service uses the variations of our negotiation method, and the other two services always use the negotiation method designed in this work.

Table 2: Performances comparison of applying DeResolver's variations to one service

Solution	Intell.	Pedestrian	Environment
DeResolver w/o opponent learning	426.61	10.83	338.90
DeResolver	172.39	20.15	378.00
DeResolver w. perfect opponent learning	164.32	20.18	565.49

The results are shown in Table 2 and there are two observations. The first one is learning the opponent services' ranking can help the service to get more benefit from the service that does not learn this information. When intelligent traffic light control service has little information on two opponent services, the other two services can reduce the average pedestrian waiting time by 46.3% and the weighted air pollutant emission by 11.5%. The second observation is perfect learning can help the service to find a more beneficial agreement, while it will decrease the performance of other services.

Table 3: Performance of DeResolver with different setting of service types

Intell.		Pedestrian		Environment	
Type	Perf.	Type	Perf.	Type	Perf.
DeResolver	160.58	Dedicated	23.17	Dedicated	397.04
DeResolver	172.39	DeResolver	20.15	DeResolver	378.00
DeResolver	185.68	Selfish	18.23	Selfish	367.93

5.3.3 Performance of DeResolver with different types of services. We show the performance of DeResolver when negotiating with different types of services in this part. First of all, we propose the definitions of service types. Selfish agent: a selfish negotiation agent is not willing to concede during the negotiation. In this experiment, we set up that a selfish agent decreases its worst acceptable configuration ranking by one every two proposing periods. For example, a selfish agent makes an proposal with ranking q at its first proposing period. It will propose another proposal with ranking $q-1$ at its third proposing period. This agent only accepts configurations with ranking no less than q at the periods between its first and third proposing period and still proposals the configuration with ranking q in its second proposing period. Dedicated agent: an agent is willing to reduce its lowest acceptable configuration ranking by two between its two consecutive proposing periods. DeResolver: the agent uses the negotiation algorithm designed in this study.

Table 4: Convergence analysis

Service type			Average # of periods to reach an agreement
Intell.	Pedestrian	Environment	
Dedicated	Dedicated	Dedicated	14.00
DeResolver	Dedicated	Dedicated	16.90
DeResolver	DeResolver	DeResolver	17.56
DeResolver	Selfish	Dedicated	22.06
DeResolver	Selfish	Selfish	28.15

Table 3 shows the performance of DeResolver with different sets of opponent services' types. The observation is that the performance of DeResolver is stable with different opponent services' types, i.e., DeResolver increases or decreases the performance by 6.9% and 7.7% with dedicated and selfish opponent services, respectively. With dedicated opponent services, DeResolver can take advantage of learning opponent services' acceptable configurations and propose the configuration which is acceptable to all services and introduces the most benefit to itself. When negotiating with selfish agents, the reactive recession strategy makes sure that the DeResolver agent does not miss too much utility according to its estimation and observation of opponents.

5.3.4 Convergence analysis with different types of services. In this part, we show the average number of periods needed to reach an agreement for three services with different sets of service types in Table 4. There are multiple observations. The first one is that this negotiation converges quickly when all agents use our negotiation method, e.g., averagely costing 17.56 periods (0.053 milliseconds on a PC) to reach an agreement with 60 possible configurations among three agents. The second one is that DeResolver is willing to concede the lowest acceptable ranking if its opponent services also concede. The third observation is if all agents use our negotiation method, each agent will concede step by step in any two consecutive proposing periods which is kind of slower compared with the case that all opponent services are dedicated agents. The last observation is that the selfish agent increases the time cost to reach an agreement since DeResolver is not willing to concede when observing selfish behaviors.

6 RELATED WORK

We organize the related work into three categories, i.e., resolving conflicts across services, automated negotiation agent design, and opponent modeling.

Resolving conflicts across services: There exist several papers on resolving conflicts across services [18, 20, 24, 25, 35]. [3] blocks the unsafe state of the target application by forcing monitor code into the app. [24], [18] and [25] resolve conflicts by assigning weight or priority to different services based on their domain and managers' understanding of each smart service. [20] suggests that alternative realizations of users' expected applications can be selected to avoid the conflicts in Internet-of-Things. These centralized solutions may experience "single point of failure" and they require city managers to have abundant knowledge of services [18, 20, 25] for determining the weight of each service [22, 24, 35]. Whereas, our decentralized negotiation-based solution does not rely on city managers or a central agent to resolve the conflicts.

Automated negotiation agent design: Multi-agent negotiation has already been widely studied in game theory [5, 8, 12]. However, they cannot be applied to solve our problem directly due to making impractical assumptions, e.g., agents' utility functions have some specified properties [15, 21], agents know complete knowledge of opponents' preference [6, 26], and there exist mediators computing agents' offers [12, 15, 21]. [15] considers finding a Pareto-efficiency solution for multi-attribute negotiation with the assumption that one mediator applies query learning to find near Pareto-efficiency solution and each agent's preference is monotonic. Our work does not rely on these assumptions to conduct negotiation automatically, which are impractical for smart city services.

Opponent modeling: Modeling opponents is essential to improve the performance of negotiation results. The closest related work to this study is learning the acceptance strategy or the preference profile of opponents [2]. To learn the acceptance strategy, existing methods focus on estimating the reservation values or the acceptance probability of different offers. An approach learns opponents' reservation value by anticipating opponents' behaviors with Bayesian learning [33] or non-linear regression [10]. Several methods are proposed to learn the preference profile. [19] uses Bayesian learning to determine the opponent types for given negotiation actions and opponent groups. All these methods make some assumptions of opponents that do not hold in this work or require some detailed information of opponents. Whereas, our solution does not make such assumptions to improve the performance of a service under negotiation.

7 CONCLUSION

Conflicts across services directly affect users' mobility and health in modern cities. To achieve dynamic resolution, we propose a decentralized negotiation and conflict resolution framework called DeResolver. Under such a framework, a learning-based solution is designed to guide how a service negotiates with other services to maximize its utility. Trace-driven simulations show that our solution achieves much more balanced results, i.e., only increasing the average vehicles' waiting time measured for intelligent traffic light control service by 6.8% while reducing the weighted air pollutant emission measured for environment control service and the pedestrian waiting time measured for pedestrian service by 12.1% and 33.1%, compared to priority-based solutions.

ACKNOWLEDGMENTS

This work was supported in part by NSF 1952096 and NSF CNS-1553273 (CAREER).

REFERENCES

[1] N. S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* (1992).
[2] T. Baarslag, M. JC Hendrikx, K. V Hindriks, and C. M Jonker. 2016. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems* (2016).
[3] Z B. Celik, G. Tan, and P. D McDaniel. 2019. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *NDSS*.
[4] F. V Cespedes, A. M Ciechanover, and M. Eiran. 2018. BreezoMeter: Making Air Pollution Data Actionable. (2018).
[5] R. M. Coehoorn and N. R. Jennings. 2004. Learning on Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs. In *ICEC '04*. ACM.
[6] U. Endriss. 2006. Monotonic Concession Protocols for Multilateral Negotiation. In *AAMAS '06*. ACM.
[7] J. Friedman, T. Hastie, and R. Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.
[8] K. Hindriks and D. Tykhonov. 2008. Opponent Modelling in Automated Multi-issue Negotiation Using Bayesian Learning. In *AAMAS*.
[9] T. K. Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*. IEEE.
[10] C. Hou. 2004. Predicting agents tactics in automated negotiation. In *IEEE/WIC/ACM IAT '04*.
[11] Intel. 2019. *Intelligent traffic management system*. https://solutionsdirectory.intel.com/solutions-directory/Intelligent_Traffic_Management_System
[12] T. Ito, H. Hattori, and M. Klein. 2007. Multi-issue Negotiation Protocol for Agents: Exploring Nonlinear Utility Spaces.. In *IJCAI*.
[13] S. Ji, Y. Zheng, Z. Wang, and T. Li. 2019. A Deep Reinforcement Learning-Enabled Dynamic Redeployment System for Mobile Ambulances. *ACM IMWUT* (2019).
[14] D. G Kleinbaum, K Dietz, M Gail, M. Klein, and M. Klein. 2002. *Logistic regression*.
[15] G. Lai, C. Li, and K. Sycara. 2006. Efficient multi-attribute negotiation with incomplete information. *Group Decision and Negotiation* (2006).
[16] L. Li, Y. Lv, and F. Wang. 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* (2016).
[17] M. Li, H. Li, and Z. Zhou. 2009. Semi-supervised document retrieval. *Information Processing & Management* 45, 3 (2009), 341–355.
[18] C. Mike Liang, B. F. Karlsson, N. D. Lane, F. Zhao, J. Zhang, Z. Pan, Z. Li, and Y. Yu. 2015. SIFT: Building an Internet of Safe Things. In *IPSN '15*. ACM.
[19] R. Lin, S. Kraus, J. Wilkenfeld, and J. Barry. 2008. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence* 172, 6-7 (2008), 823–851.
[20] R. Liu, Z. Wang, L. Garcia, and M. Srivastava. 2019. RemedioT: Remedial Actions for Internet-of-Things Conflicts. In *BuildSys '19*. ACM.
[21] Y. Lou and S. Wang. 2016. Approximate representation of the Pareto frontier in multiparty negotiations: Decentralized methods and privacy preservation. *European Journal of Operational Research* 254, 3 (2016), 968–976.
[22] M. Ma, S. M. Preum, and J. A Stankovic. 2017. Cityguard: A watchdog for safety-aware conflict detection in smart cities. In *IoTDI*. ACM.
[23] M. Ma, S M. Preum, W Tarneberg, M. Ahmed, M. Ruiters, and J. Stankovic. 2016. Detection of runtime conflicts among services in smart cities. In *SMARTCOMP*.
[24] M. Ma, J. A Stankovic, and L. Feng. 2018. Cityresolver: a decision support system for conflict resolution in smart cities. In *ICCPS '18*.
[25] S. Munir and J. A Stankovic. 2014. Depsys: Dependency aware integration of cyber-physical systems for smart homes. In *ICCPS '14*.
[26] J. F Nash Jr. 1950. The bargaining problem. *Econometrica: Journal of the Econometric Society* (1950), 155–162.
[27] NYC. 2020. *NYC Open Data*. https://opendata.cityofnewyork.us/
[28] City of Newark. 2020. *City of Newark Open Data*. http://data.ci.newark.nj.us/
[29] Graz University of Technology. 2019. *New traffic light system automatically recognizes pedestrians' intent to cross the road*. Retrieved Nov 24, 2019 from https://phys.org/news/2019-05-traffic-automatically-pedestrians-intent-road.html
[30] R. K. Pasumarthi, S. Bruch, X. Wang, C. Li, M. Bendersky, Ma. Najork, J. Pfeifer, N. Golbandi, R. Anil, and S. Wolf. 2019. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *KDD '19*. ACM.
[31] A. Piscitello, F. Paduano, A. A Nacci, M. D Noferi, D.and Santambrogio, and D. Sciuto. 2015. Danger-system: Exploring new ways to manage occupants safety in smart building. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*.
[32] R. S Sutton and A. G Barto. [n.d.]. *Reinforcement learning: An introduction*.
[33] K. Sycara and D. Zeng. 1997. Benefits of learning in negotiation. In *AAAI '97*.
[34] M. Szummer and E. Yilmaz. 2011. Semi-supervised learning to rank with preference regularization. In *CIKM '11*. ACM.
[35] R. M. B. S. Thais, B. R. Linnyer, and A. F. L. Antonio. 2010. How to conciliate conflicting users' interests for different collective, ubiquitous and context-aware applications?. In *IEEE Local Computer Network Conference*. 288–291.
[36] S. Wang, T. He, D. Zhang, Y. Shu, Y. Liu, Y. Gu, C. Liu, H. Lee, and S. H. Son. 2018. BRAVO: Improving the Rebalancing Operation in Bike Sharing with Rebalancing Range Prediction. *ACM IMWUT* (2018).
[37] F. V. Webster. 1958. *Traffic signal settings*. Technical Report.
[38] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li. 2019. Colight: Learning network-level cooperation for traffic signal control. In *CIKM '19*.
[39] H. Wei, G. Zheng, H. Yao, and Z. Li. 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In *KDD*.
[40] M. Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *ICML'2000*. 1151–1158.
[41] H. Yang, S. Tsai, K. Liu, S. Lin, and J. Gao. 2019. Patrol Scheduling Against Adversaries with Varying Attack Durations. In *AAMAS '19*.
[42] Y. Yuan, D. Zhang, F. Miao, J. Chen, T. He, and S. Lin. 2019. p^ 2Charging: Proactive Partial Charging for Electric Taxi Systems. In *ICDCS*.
[43] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He. 2013. CoRide: Carpool Service with a Win-Win Fare Model for Large-Scale Taxicab Networks. In *SenSys '13*.

A APPENDIX

A.1 General Formulation of DeResolver

We propose that multiple services that result in a conflict can play a negotiation to resolve this conflict, which is the main idea of DeResolver. In this section, we provide a general mathematical formulation of DeResolver, which shows how to define and formulate a negotiation for a conflict across services.

Negotiation agent: A negotiation is organized for resolving a conflict, and it consists of N services whose control decisions result in a conflict. For example, if N services have inconsistent configurations of an actuator, these services play a negotiation to resolve the inconsistency. A negotiation agent represents a service. These N services negotiate the issue under discussion with a time horizon of H periods.

Proposal: It is a tentative suggestion about a solution to the issue under discussion. Let O_i^h be the proposal that is made by service i to the other $N-1$ services during a negotiation period h. In a negotiation, the issue under discussion can be a configuration of an actuator for a direct conflict, or the distribution of a shared common resource, e.g., the upper bound of noise and the air pollutant emission budget, for an environmental conflict.

Agreement: We define that all the negotiation agents (N services) achieve an agreement if there exists a proposal O_i^h that is accepted by the other $N-1$ services.

Utility: It represents the benefit that a service i can receive by applying a proposal $O_{i'}^h$, denoted as $r_i(O_{i'}^h)$. The utility function is defined according to the objective of each service i. For example, this function may represent the number of packages that are delivered on time for package delivery service, or the inverse value of total vehicle waiting time for intelligent traffic light control service.

Multi-agent negotiation protocol: A key issue in designing a negotiation among multiple services is to determine a protocol that these services obey. The negotiation protocol for resolving a conflict is defined as follows.

The negotiation process terminates if N negotiation agents reach an agreement within the H time periods, or they cannot find an agreement after H time periods. The length of a time period can be set as a static value, e.g., a second. The maximum number of time periods (H) is determined according to the specific application scenario. We demonstrate how to set H for the Example 2.1 in Section 3.

During the negotiation, N services make their proposals by the round-robin principle during different negotiation periods. The round-robin principle means only a service proposes its solution to the issue under discussion during a negotiation period and N services make the proposals in a circular order. If a service makes a proposal during period h, it should make another proposal during the negotiation period $h+N$ as long as neither an agreement is reached nor the negotiation terminates. If there exists a negotiation agent that rejects the proposal O_i^h, the negotiation moves to the period $h+1$, and another service makes its proposal.

During the negotiation, any service i knows which service makes what proposal during any negotiation period h, and the responses from other $N-1$ services, i.e., acceptance or rejection. However, service i does not know the opponent services' objectives. Service i can only observe the behaviors of other services during the negotiation.

A.2 State of a city

The three services list in Example 2.1 are interested in the different state of the city. We list them as follows.

- Intelligent traffic light control service: the state component includes the number of waiting vehicles, the vehicle arriving rate in each direction, the vehicle throughput of each direction, the updated waiting time of vehicles, and the states of the traffic light in current phase t and next phase $t+1$.
- Pedestrian service: the state component includes the number of waiting pedestrians, the pedestrian arriving rate, and the pedestrian throughput of each direction, the updated waiting time of pedestrians, and the state of the traffic light in current phase t and next phase $t+1$.
- Environment control service: the state is defined as the combination of the number of vehicles on the adjacent road segments, the number of waiting vehicles $V_{l''}$, the vehicle arriving rate and the throughput of each direction, and the state of the traffic light in current phase t and next phase $t+1$.

A.3 Utility Function

We list the utility function of three services in Example 2.1 as follows.

Intelligent traffic light control service: the objective is to minimize the total waiting time of vehicles around the intersection k, where waiting vehicles include taxis, bikes, buses, and private cars. Let i be 1 to represent this service and the immediate utility function is: $r_{1,k}(O^h_{i',k}(t)) = -\sum_{t'=1}^{O^h_{i',k}(t)} \sum_{l\in I_k} W_{1,l}(t')$. I_k is the set of approaching lanes of intersection k. t' is a time slot of phase t, and a phase consists of several time slots, e.g., a time slot is one second and there are five time slots in a phase. $W_{1,l}(t')$ are the total waiting time of waiting vehicles in approaching lane l at time slot t'. The inner sum represents all vehicles' waiting time by the end of slot t' and the outer sum is the sum of all vehicles' waiting time over the phase t. To minimize the waiting time, additive inverse of total waiting time is used when maximizing the immediate utility.

Pedestrian service: its objective is similar with that of intelligent traffic light control service, stated as minimizing the waiting time of pedestrians in a road intersection k. Then the utility $r_2(O^h_{i',k}(t))$ of applying configuration $O^h_{i',k}(t)$ at intersection k for pedestrian service is formulated as: $r_{2,k}(O^h_{i',k}(t)) = -\sum_{t'=1}^{O^h_{i',k}(t)} \sum_{l'\in I'_k} W_{2,l'}(t')$. I'_k denotes the set of pedestrians' walking directions of road intersection k and $W_{2,l'}(t')$ are the total waiting time of waiting pedestrian in direction l' at time slot t'. We also maximize the additive inverse.

Environment control service: this service aims at optimizing the environment quality, e.g., decreasing the noise level and air pollutant emission in all road segments. For given a time slot t' during traffic light phase t and one road segment l'', let $f(V_{l''}(t'))$ denote the value of environment quality, where $V_{l''}(t')$ is the number of vehicles on road segment l'' during time slot t' that connects with road intersection k. Then the immediate utility of environment control service is formulated as: $r_{3,k}(O^h_{i',k}(t)) = -\sum_{l''\in I''_k} \omega_{l''} \times f(V_{l''}(t'))$. $\omega_{l''}$ is the weight for road segment l'' that can be defined by the environment around each road segment, e.g., a road segment has high weight if there are hospitals or schools around it. I''_k is the set

of adjacent road segments for intersection k and $f(\cdot)$ is a function calculating the environment quality for given number of vehicles.

A.4 Proof of Theorem 4.1

Theorem A.1. *Consider N services in a negotiation, they negotiate to determine a configuration from the configuration space $D = \{T_{min}, T_{min}+1, \cdots, T_{max}\}$. If all N services use our negotiation algorithm, and the estimation of acceptable configurations and their ranking to any service i is accurate, the result of our algorithm is guaranteed to reach a Pareto-optimal agreement.*

Proof. Suppose there are N services playing a negotiation during period $1, 2, \cdots, H$. The period from $N(r-1)+1$ to rN is called round r. All services take turns to propose an action in each round until they reach an agreement. Let $A^r_{i,i'}$ be the set of acceptable configurations of service i' that is estimated by service i in round r. Specially, $A^r_{i,i}$ is service i's true acceptable configuration when $i = i'$.

According to our algorithm, at any round r service i first expands its acceptable configuration $A^r_{i,i}$ with a lower utility, that is to say,

$$A^r_{i,i} = A^{r-1}_{i,i} \cup c^r_i, \tag{4}$$

where $c^r_i < min\{A^{r-1}_{i,i}\}$. Then it computes

$$C = \bigcap_{i=1}^{N} A^r_{i,i}. \tag{5}$$

If $C \neq \emptyset$, then service i selects the candidate introducing the largest utility from $C \cap Pe$, where Pe is the set of Pareto-optimal solutions that are calculated by the estimated ranking of configurations for other services i'; if $C = \emptyset$, then c^r_i is service i's proposal at round r.

Notice that $|A^r_{i,i}| > |A^{r-1}_{i,i}|$ based on (4), also we have $|A^r_{i,j}| > |A^{r-1}_{i,j}|$ because all services use our negotiation algorithm. Therefore, $\exists r \forall j, |A^r_{i,j}| = |D|$. In this case, $C \neq \emptyset$, because $\forall j, A^r_{i,j} = D$. In other words, all services must reach an agreement at round r.

Finally, we show that the agreement is a Pareto-optimal solution. Under the assumption that both the estimation of acceptable configurations and ranking of configurations are accurate, the proposal determined by $\text{argmin}_{d \in A^r_{i,i}} U_i(d)$ is not accepted by the other $N-1$ services. The reason is $C = \emptyset$. Only the proposal generated by $d = \text{argmax}_{d \in C \cap Pe} U_i(d)$ can be an agreement, and the agreed proposal is Pareto-optimal since it is an element of the set of Pareto-optimal agreements according to $d \in C \cap Pe$. □

A.5 Action-utility Table Computation

It is essential for any service i to know the utility that different configurations can introduce. Even without conflicts, each service also needs to compute such a table to choose the optimal configuration for optimizing the quality of service. The action-utility table computation is beyond the scope of this study, i.e., addressing the conflicts across services.

For the traffic light control example, determining optimal control actions has already been well studied in the previous work, classified into two categories: conventional methods and reinforcement learning based solutions. Conventional methods [6, 37] configure fixed schedule or changing rules according to previous knowledge, which are vulnerable to the dynamic traffic condition. Reinforcement learning based methods [16, 39, 40] take real-time traffic conditions as input, and aim at selecting the action resulting in the maximum reward. Based on the related work [39], we design a reinforcement learning based agent to control any traffic light k.

The state, action and reward (utility) of a RL agent for three services are defined in Section 3 respectively. Given the real-time state, the task of an agent is to find the action (length of the next traffic light phase) that maximizes the long-term reward, following the Bellman Equation [32]: $U_{i,k}(s_{i,k}(t), a) = r_i(a) + \gamma \max U_{i,k}(s_{i,k}(t+1), a')$. s_t is the state of the city used by service i at the beginning of traffic light phase t. The long-term action reward is the summation of the reward of the next traffic light phase $t+1$ and the maximum potential future reward.

A.6 Definition of Estimation Accuracy

We define the estimation accuracy as the main metrics for evaluating the learning based algorithm. The accuracy of estimating the acceptable or unacceptable configurations to service i is used to measure the performance of the first level of the learner. The metric is defined as: $Acc^1_i = \sum_{m=1}^{M} \sum_{h=1}^{H_m} NC_{i,m,h} / \sum_{m=1}^{M} \sum_{h=1}^{H_m} C_{i,m,h}$. M is the number of negotiations that are organized during the evaluation. H_m is the number of periods that last in the negotiation m. $NC_{i,m,h}$ is the number of configurations that are correctly classified as acceptable or unacceptable to service i in the period h of m-th negotiation. $C_{i,m,h}$ is the number of configurations that are classified as acceptable or unacceptable to service i in the period h of m-th negotiation. The accuracy of estimating how service i ranks the acceptable configurations is used to measure the performance of the second level of the learner. The metric is defined as: $Acc^2_i = \sum_{m=1}^{M} \sum_{h=1}^{H_m} NP_{i,m,h} / \sum_{m=1}^{M} \sum_{h=1}^{H_m} P_{i,m,h}$. $NP_{i,m,h}$ is the number of pairs of configurations, whose ranking order is estimated correctly in the period h of m-th negotiation. $P_{i,m,h}$ is the number of pairs of configurations, whose ranking order is estimated in the period h of m-th negotiation. The accuracy of a learner shows how often a service correctly estimates its opponent services' behaviors, which is useful for the service to conduct negotiation efficiently.