Proceedings of the Second Workshop on Synergies Between Multiagent Systems, Machine Learning and Complex Systems (TRI 2015)

held together with IJCAI 2015
Buenos Aires, Argentina

July 27, 2015

Preface

This volume contains the papers accepted at TRI 2015, the Second Workshop on Interfaces between Multiagent Systems, Machine Learning and Complex Systems held on July 26 2015 in Buenos Aires.

This second edition of the TRI workshop was co-located with the International Joint Conference on Artificial Intelligence 2015 (IJCAI-15).

The TRI workshop series aims at investigating the existing and further synergies between the Multi-agent Systems (MAS), Machine Learning (ML) and Complex System (CS) disciplines. MAS can efficiently manage domains with distributed data and expertise. Also, they have the ability to solve large and complex problems. The expertise of each agent can focus on specific computational intelligence models such as learning classifiers, evolutionary algorithms, swarm intelligence techniques, or other specific optimization/learning algorithms. Applications range from bioinformatics and traffic networks to information retrieval and text classification. Broadly speaking, the area of CS is grounded on similar ideas: complex systems investigate how relationships between components of a system give rise to emerging collective behaviors and how the system as a whole and its components interact with an environment. ML has being used both in MAS as well as in CS when the agents or components of the system need to learn to make decisions. Moreover, since MAS and CS are becoming large and more and more complex, ML is key to improve performance. Here, not only the traditionally applied method of reinforcement learning shall be studied, but there is potential for using supervised and non-supervised ML techniques as well. Since these synergies among the three areas are not well studied, we see room for such a workshop where researchers and practitioners from the three areas could come together, as currently these three communities do not meet often.

The main goals of the TRI 2015 workshop series are:

1. To bring together researchers working on Multi-agent Systems, Machine Learning, and Complex Systems
2. To explore the bridges among these disciplines in order to create new holistic knowledge from their interactions.
3. To produce contributions that allow the research community to create new challenging results and real world applications.

In particular, this workshop edition aimed at addressing the following topics:

- Adaptation and distributed learning of MAS in complex networks
- MAS Adaptability to cooperate and collaborate in the Internet of Things
- MAS and learning applied to Big Data
- Adaptive agents and MAS applied to Cloud Computing
- Emergent behavior in adaptive multi-agent systems
- Adaptive agents and MAS for self-organizing complex systems
- Bio-inspired multi-agent systems

- MAS and learning applied to Bioinformatics
- Adaptive MAS for simulating real world complex scenarios
- Adaptive agents and e-commerce: negotiation, trust models, reputation, coordination, etc.
- Game theoretical analysis of adaptive multi-agent systems over complex networks
- Adaptive MAS in Economics: microeconomics, macroeconomics, global markets, etc.
- Adaptive Multi-agent Systems for Smart Grids and Smart Cities
- Adaptive MAS for Urban Transport Control and Transportation
- Application of supervised and non-supervised Machine Learning in MAS
- Application of supervised and non-supervised Machine Learning in complex systems
- Learning for mechanism design in complex systems
- Community detection in MAS
- Learning in the context of social networks
- Microscopic and agent-based simulation of complex systems (e.g. using cellular automata and other techniques)

This volume includes the six papers accepted for oral presentation. Each submission was reviewed by at least three members of the program committee.

The program also includes two invited talks given by Prof. Sarvapali Ramchurn (University of Southampton) and Dr. Yair Zick (Carnegie Mellon).

The organising committee would like to thank all the authors of submitted papers for their submissions and all the reviewers for the quality of their reviews.

We would also like to thank the IJCAI 2015 organisers for hosting the TRI workshop, and particularly Jerôme Lang, the IJCAI 2015 workshop chair.

July 8, 2015
Vigo

Ana L. C. Bazzan
Juan Carlos Burguillo
Juan Antonio Rodriguez
Aguilar

Program Committee

Stefania Bandini	University of Milano-Bicocca
Tiago Baptista	Universidade de Coimbra
Ana L. C. Bazzan	Universidade Federal do Rio Grande do Sul
Christian Blum	University of the Basque Country
Juan Carlos Burguillo	University of Vigo
Didac Busquets	Imperial College
Aleksander Byrski	AGH Krakow
Camelia Chira	Technical University Cluj Napoca
Roberto Da Silva	Universidade Federal do Rio Grande do Sul
Giovanna Di Marzo Serugendo	University of Geneva
Jelena Fiosina	TU Clausthal
Maksims Fiosins	TU Clausthal
José Fernando Fontanari	Universidade de São Paulo
Marie-Pierre Gleizes	IRIT, Toulouse
Nathan Griffiths	University of Warwick
Rashedul Hasan	U. of Nebraska
Joanna Kolodziej	Cracow University of Technology
Zhao Liang	Universidade de São Paulo
Pedro Mariano	Universidade de Lisboa
Peter McBurney	King's College London
José Manuel Molina	Universidad Carlos III de Madrid
Enrique Munoz de Cote	INAOEP
Juan Pavón	Universidad Complutense de Madrid
Steve Phelps	University of Essex
Juan Antonio Rodriguez	IIIA-CSIC
Andrea Roli	University of Bologna
Frank Schweizer	ETH Zürich
Joan Serrà	IIIA-CSIC
Kagan Tumer	Oregon State University
Karl Tuyls	University of Liverpool
Ivan Zelinka	University of Ostrawa

Table of Contents

Artificial Prediction Markets for Online Prediction of Continuous Variables

Fatemeh Jahedpari[1], Marina De Vos[1], Sattar Hashemi[2], Benjamin Hirsch[3], Julian Padget[1]

[1] Department of Computer Science, University of Bath, UK
[2] Computer Science and Engineering Department, Shiraz University, Iran
[3] EBTIC, Khalifa University, United Arab Emirates

Abstract. We propose the Artificial Continuous Prediction Market (ACPM) as a means to predict a continuous real value, by integrating a range of data sources and aggregating the results of different machine learning (ML) algorithms. ACPM adapts the concept of the (physical) prediction market to address the prediction of real values instead of discrete events. Each ACPM participant has a data source, a ML algorithm and a local decision-making procedure that determines what to bid on what value. The contributions of ACPM are: (i) *adaptation* to changes in data quality by the use of learning in: (a) the market, which weights each market participant to adjust the influence of each on the market prediction and (b) the participants, which use a Q-learning based trading strategy to incorporate the market prediction into their subsequent predictions, (ii) *resilience* to a changing population of low- and high-performing participants. We demonstrate the effectiveness of ACPM by application to an influenza-like illnesses data set, showing ACPM out-performs a range of well-known regression models and is resilient to variation in data source quality.

1 Introduction

Physical world prediction markets aim to utilise the aggregated "wisdom of the crowd" to predict the outcome of a future event [13], such as who will win an election. In these markets, participants buy and sell instruments, called securities, whose payoffs are tied to the occurrence of the specified future event. A prediction market is run by a market-maker who interacts with traders to buy and sell the securities. Artificial Continuous Prediction Market (ACPM) adapts the concept for the purpose of predicting a real value in a continuous domain. Our motivation in developing ACPM is to use online learning in situations in which it is desirable to integrate data dynamically from a variety of sources whose data quality is (time-)variable, using a variety of analysis algorithms.

A prediction market is created for each prediction that a participant can make based on the data in their streams. All the data needed for this, including the correct prediction, is referred to as record in accordance with the ML literature. The participants, which we refer to as agents, predict the value of the record using data from their assigned source and their analysis algorithm. Subsequently, the market maker calculates the market prediction by combining all the individual predictions. Once the true value of the record is known, the market maker computes the reward for each agent and informs the agents about the outcome so they can update their analysis algorithm and their trading strategy, with the aim of improving future market predictions.

We use a series of experiments over an influenza-Like Illness (ILI) dataset to show how ACPM can effectively be applied to the problem of syndromic surveillance. The main objective and challenge of a syndromic surveillance system is the earliest possible detection of a disease outbreak within a population. Much research has been done to discover potential data sources and alternative analysis algorithms for each data source in the syndromic surveillance domain [2]. An issue with syndromic surveillance data sources is that data quality fluctuates over time. For example, Google Flu Trends may show false alerts as a result of a sudden increase in ILI related queries due to unusual events, such as a drug recall for a popular cold or flu remedy [5]. Therefore, integrating available data sources according to an adaptive weighting scheme over time seems necessary. In addition, given that the quality of data changes over time, and the most suitable algorithm for a given data source is not necessarily known *a priori*, a reasonable response is to analyse each data source with a variety of algorithms and integrate their results.

In the experiments, we predict the level of ILI activity for a specific date in a certain region using ACPM to integrate the various data sources, analysed by different algorithms. We show that the system performs at least as well as all the market participants and adding learning to the agents' trading strategy improves market prediction. The results also highlight that ACPM outperforms well-known regression models and ensembles, that are commonly used for this type of reasoning. The rest of the paper is organised as follows. Section 2 explains the details of ACPM. Section 3 evaluates our model and analyses the results. Section 4 covers related work and concludes.

2 ACPM Description

2.1 Overview

ACPM is an online machine learning technique which adapts the concept of a (physical) prediction market to populate it with artificial agents as market participants[4]. We assume participants are benevolent and self-interest is not an issue, which means they are not competitive and they work together to get the best outcome of the system. Each participating agent receives information from its designated data source and analyses its data with its given analysis algorithm. Each ACPM also includes a market maker who runs the market, deals with agent transactions and establishes the market prediction.

The market maker instantiates a prediction market for each record with the purpose of predicting its true value. Each market comprises a number of rounds, where each agent sends its bids to the market maker. Each bid comprises: (i) a prediction value which, in our case study would be the number of cases of flu in the USA for a certain week of the year and (ii) the amount the agent is betting on its prediction. Each agent, using the data for that record and its accumulated knowledge, analyses the data and predicts the true value of the record. Then, based on its trading strategy and its (available) capital, it determines how much to invest. Once a round is completed, the market

[4] The terms participating agent and agent are used interchangeably.

maker announces the market prediction based on bids received and an agent can use this information to update its bid via its trading strategy in subsequent rounds. The market maker then seals the bid in the last round, i.e deducts capital from the agent according to its bid, rewards agents and reports the final market prediction. In this way, the period between the first round and the last round can be used to train the agents to increase their prediction accuracy based on the integrated predictions of other participants.

Once the market is over, agents are notified of the correct answer (the true value of the record) and receive an amount of revenue as determined by a reward function. Each agent learns from each market, based on the revenue they receive and the losses they make, in addition to finding out the correct answer. Consequently, they can, if desired, update their strategy, analysis algorithm and beliefs for future markets. Agents learn by updating their analysis algorithm with the correct answer for the record and updating their trading strategy based on how much they could earn if behaving differently (as explained in Section 2.5). The market maker learns indirectly through updating the agents' capital. Their capitals determine their bidding power and hence the weight of their prediction. The market maker integrates agent predictions using an integration function and rewards agents based on a reward function. In our continuous variable prediction setting, the existing discrete existing Market Scoring Rule (MSR) technique [6] is not suitable for our system. In the next sections, we propose our continuous versions.

2.2 Integration Function

At the end of each round, the market maker uses an integration function to decide the market prediction, based on the received bids. We use the following formula:

$$\text{Market Prediction} = \frac{\sum_{b=1}^{n} Prediction_b * Invest_b}{\sum_{b=1}^{n} Invest_b} \tag{1}$$

where n is the number of bids

This formula assigns more weight to predictions backed by higher investments. Participants who accrue more capital, due to their success in earlier markets, have the opportunity to invest more and so get greater influence in the market.

2.3 Reward Function

At the last round, agents are notified of the correct answer and receive revenue as determined by a reward function. These revenues are added to their capital. The reward an agent receives is inversely proportional to the agent's prediction error, thus incentivising accurate prediction, making our reward function incentive compatible. Equation 2 describes a family of reward functions, where different values of $P \in \mathbb{R}^+$, $\beta \in \mathbb{R}^+$ and $C \in \mathbb{R}^+$ result in the curves shown in Figure 1, in which $P \geq 1$ generates convex functions (above diagonal) and $0 < P < 1$ generates concave functions (below diagonal).

$$Reward = \max(\beta * \frac{-1}{C^P} * error^P + \beta, 0) \tag{2}$$

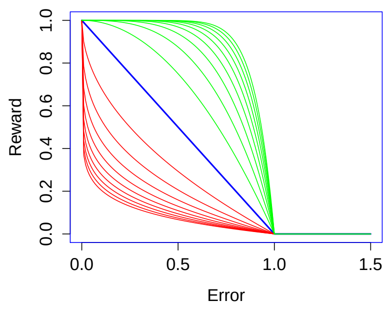


Fig. 1. Reward functions. In this figure $C = 1$, $\beta = 1$ and each of the curve lines refers to one value of $P \in (\frac{1}{10}, \frac{1}{9}, ...\frac{1}{2}, 1, 2...10)$.

where $error = abs(TrueValue - AgentPrediction)$. The actual revenue accrued by an agent is the product of its reward and the amount invested on its prediction. Thus, the more an agent invests, the more revenue it receives. Consequently, agents with higher confidence are incentivised to invest more and hence have a greater influence on the market. In addition, the agents with low capital (indicating low past performance) cannot invest and influence the market prediction as much as high performing agents, who acquire more capital over time.

Coefficient C determines the reward cut-off, above which agents receive zero. As can be seen in Figure 1, the reward function is flat and equal to zero after the cut-off (C = 1). The slope of the reward function differentiates between the participating agent rewards in proportion to the error in their predictions. Increasing C, while keeping the other parameters fixed, decreases the slope of the reward function, and consequently decreases differentiation. Conversely, increasing C increases the number of agents that receive rewards. An agent's error is computed relative to the correct answer for a given market. As the range of an agent's error may change from market to market (as they learn) and from domain to domain, the cut off cannot be a fixed value, but rather be calculated for each market so that a specific percentage of agents receive positive rewards. For example, C can be calculated for each market to be equal to the maximum error of all agents in that market. Intuitively, a certain amount of differentiation is desirable and lower or higher values of that could harm the performance of the system. For example, high differentiation means that a few high quality agents lead the market and predictions of the majority of agents, including good quality ones, can be under-weighted. On the other hand, low differentiation narrows the gap between the influence of high and low quality agents, so that insufficient account is taken of the more accurate agents.

Coefficients P and β shrink (or enlarge) the function horizontally and vertically respectively. Increasing P increases the degree of curvature of the reward function, and consequently, decreases the differentiation among agents especially those with low errors. Increasing β has the effect of a linear increase in both agent revenue and in differentiation between participants. With $\beta = 1$, an agent loses a fraction of its money according to the error they make and only in the best case, where the error is zero, do they neither earn nor lose. This value disincentives participation, since return is less than investment and the steady depletion of capital leads to their holding very little in later markets. The default values of P=1 and $\beta = 2$ generate a simple linear reward function which has the property of being incentive compatible.

2.4 Rate Per Transaction

The system has two other parameters: Maximum Rate Per Transaction (MaxRPT) and Minimum Rate Per Transaction (MinRPT), which specify the maximum and minimum percentage of the capital that each participant can invest. The purpose of the MaxRPT parameter is to prevent unsuccessful agents bankrupting themselves and being eliminated from the market. It is not desirable to reduce the population, because that leads to the loss of a data feed or the loss of an analysis algorithm: while qualitatively low at some point, the combination might improve again over time. The MaxRPT parameter can be used to tune system response to the degree of environment volatility. For example, in situations where the quality of agents' data fluctuates frequently, MaxRPT should be high so that an affluent agent loses most of its capital if its error is high for a few successive markets. On the other hand, MaxRPT should be low in situations where we expect that the quality of good agents remains good even though they may make occasional mistakes. If $MaxRPT < 1$, an agent's capital may get very small but is not used up entirely. Hence it can invest and recover at any time, albeit slowly!. The purpose of MinRPT is to prevent the system from being unresponsive in cases where none of the participating agents have enough incentive to invest.

2.5 Agent Trading Strategy

As mentioned earlier, agents can use the market prediction, received from the market maker at the end of each round, to update their bids for subsequent rounds. In this paper, we examine two strategies: a constant one and a Q-Learning based one.

Constant Strategy: Agents simply dedicate a fixed ratio of their capital to bid in each round. In this paper, this percentage is equal to MaxRPT. This naïve strategy ignores the advantage of updating the prediction on the basis of the market prediction of the previous round.

Q-Learning Trading Strategy: In reinforcement learning, agents explore their environment and learn to choose actions that maximise their rewards. Agents are seen as finite state machines. They receive a reward for the action they take to reach another state. In the Q-learning algorithm [17], agents have a state action value function $Q(s, a)$ which estimates the expected reward for performing an action a in state s. A greedy policy suggests choosing the action that gives the highest expected reward in the given state.

In our Q-learning based trading strategy, agents recognise their state by (i) measuring the difference between their prediction and the market prediction of the previous round, (ii) the current round number. Here, we have just two actions. The difference between these two actions is whether the agent use the market prediction as another source of information or not to change its prediction. While the first action (PreservePr) suggests the agent ignores the market prediction of the previous round, the second one

(ChangePr) suggests the agent shifts its prediction linearly by a percentage, called δ, towards the market prediction.

In both actions, the agent uses a simple betting strategy which assumes that the correct answer is equal to the market prediction of the previous round. Based on this assumption and its prediction value, as calculated by its analysis algorithm, the agent estimates its error which is absolute difference of agent prediction and market prediction. Then, the agent uses the estimated error and the reward function setting, which was used by the market maker in the previous market, to estimate its expected reward. If the expected reward is less than one, which means that the agent earns less than what it invests, then the betting strategy suggests the agent invests MinRPT percentage of its capital, and otherwise MaxRPT of the capital[5].

Agents update their state action value function once the market is over and the correct answer is revealed. Each agent revises all States s, which it was confronted with during the market period. The agent assigns the state action values for each Action a in State s equal to the the amount of net revenue – its revenue minus the investment amount – it could obtain by performing Action a in State s. The agent also calculates and stores what was the best value of δ for state s. Formula 3 linearly calculates the percentage the agent should shift its prediction towards the market prediction, with a limit of 100 percent.

$$\delta = \min(abs(\frac{\text{correct answer} - \text{agent prediction}}{\text{market prediction} - \text{agent prediction}}), 1) * 100\% \tag{3}$$

In the first market, as the agent's knowledge is void, the agent just bids the MinRPT percentage of its capital. In all other markets, the agents have no information about the market prediction in the first round, therefore they use the constant strategy. In all other rounds, agents use a greedy strategy which means they refer to their state action value function and choose the action with the highest state action value.

3 Evaluation

We evaluate the performance of ACPM by applying it to syndromic surveillance in the USA. In this context, the system predicts the disease activity level of influenza-like illnesses (ILI) in a given week in the whole of the USA using publicly available data sources. The data used here contains more than 100 real data streams covering the period 4th January 2004 to 27th April 2014, from a variety of sources including Google Flu Trends (GFT), Centers for Disease Control and Prevention (CDC), Google Trend.

We have used weekly Google Flu Prediction for different areas of the United States including states, cities and regions for which GFT data is available since 2004. Google Trend statistics for different terms such as “flu”, “fever cough sore throat”, “flu symptoms” and CDC statistics[6] including CDC ILI rate for different age groups, USA na-

[5] Two other models of betting strategy were tried, but this one both maximises system performance and agent utility.

[6] CDC reports ILI rates with a two-week time lag. Therefore, in order to align CDC data with the other data streams used, we take the ILI rate from two weeks earlier for each week of the experiment period.

tional ILI rate, total number of patients and total number of outpatient healthcare providers in ILI network are used[7]. The ACPM prediction is compared against the CDC ILI rate.

We refer to data streams as having low, medium or high quality, based on their mean absolute error (MAE) as reported by several regression models. These categories are not absolute judgements, but relative ones confirmed through the use of several classifiers in order to cluster the data streams according to their mean absolute error (MAE) and hence identify threshold values that fall between the clusters.

3.1 Hypotheses

Using two sets of experiments, we evaluate ACPM against the following hypotheses:
H1: ACPM performance is higher than its best performing agent.
H2: ACPM is resilient to different proportions of low- and high-performing participants.
H3: Adopting the Q-learning trading strategy, compared to the constant strategy, improves ACPM performance.
H4: The Q-learning trading strategy encourages low quality agents to change their prediction based on aggregated prediction of other agents.
H5: The Q-learning trading strategy encourages high quality agents to ignore market prediction as another source of information.
H6: ACPM outperforms well-known regression models
and ensembles.
H7: Adopting Q-learning based trading strategy improves each participating agent's performance.

3.2 Set 1

The first group of experiments explores the impact of data quality on ACPM's predictive capability.

Settings For these experiments we look at four different market types (Table 1) with different proportions of participant data stream quality. Market type 1 comprises only agents with medium quality data. In order to investigate how the presence of a small number of low and high quality agents affect ACPM performance, market type 2 comprises mostly medium and a few high quality data agents and market type 3 contains mostly medium and several low quality data agents. Market type 4 contains all three kinds (a small number of low and high quality and many medium quality data agents). Each market type has 100 agents.

In these experiments, all agents use the Q-learning trading strategy and, randomly selected, analysis algorithm, namely SGD [8] algorithm. There is no specific reason for the use of SGD: it is just one of the several used for the initial clustering. The effective values of market parameters, as discussed in Section 2, can experimentally be chosen by measuring the performance of the system on historical records. Experiments gave

[7] These data can be accessed from http://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
[8] SGD loss function is set to Squared Loss function for the purpose of performing regression.

		Low Data Quality Agents		Medium Data Quality Agents		High Data Quality Agents	
Market Type	Average Error (Variance)	Quantity	Average Error (Variance)	Quantity	Average Error (Variance)	Quantity	Average Error (Variance)
Type 1	0.6043 (0.0007)	0	-	100	0.6042 (0.0007)	0	-
Type 2	0.6009 (0.0009)	0	-	97	0.6033 (0.0008)	3	0.5243 (4.79002E-05)
Type 3	0.6214 (0.0030)	12	0.7423 (0.0020)	88	0.6048 (0.0008)	0	-
Type 4	0.6198 (0.0033)	13	0.7406 (0.0019)	84	0.6044 (0.0008)	3	0.5243 (4.79002E-05)

Table 1. Our four market types.

us: (i) number of rounds $= 2$, (ii) MaxRPT$= 0.9$, (iii) MinRPT$= 0.001$, (iv) $P = 7$, (v) $\beta = 4$, and (vi) C is chosen so that 60% of agents receive positive rewards.

Experiments The first experiment (Figure 2), compares the MAEs of the system and the best performing participant for each market type. Next, (Figure 3), we compare, for each market type, the MAE of ACPM where participants use Q-learning with one where participants do not. In the last experiment of this set, as displayed in Figure 4, we compare the use of the Q-learning actions for each agent-type in a type 4 market.

Results These experiments indicate that, as shown in Figure 2, the system's MAE is less than the best agent's MAE, without manipulating its prediction using Q-learning strategy, for every market type. The error bars show the standard error when calculating the mean absolute error. Experiments are run once as they are deterministic. Figure 3 shows that adopting the Q-learning reduces the MAE compared to the constant trading strategy in each market type (P-value [9] < 0.05 for all market types except Type 3).

As can be seen from Figure 4, Action PreservePr which suggests the agent not change its prediction, based on the previous round market prediction (as discussed in Section 2.5), is the most popular action in agents with high quality data and the least popular action in agents with low quality data. Conversely, Action ChangePr which suggests the agent change its prediction by rate δ, based on the previous round market prediction, is the most popular action in agents accessing low quality data and the least popular action in agents accessing high quality data.

3.3 Set 2

The next group of experiments compares ACPM with well-known regression models and ensembles.

Settings In this set of experiments, the market includes 14 participants, each agent has access to all 100 data streams of type 4 market, described in Table 1. Each agent uses one of the following regression models : SGD, IBK, LinearRegression, SMOreg, REP-Tree, ZeroR, DecisionStump, SimpleLinearRegression, DecisionTable, LWL, Bagging, AdditiveRegression, Stacking and Vote as its analysis algorithm. The market runs for

[9] The null hypothesis is that the two accuracies compared are not significantly different.

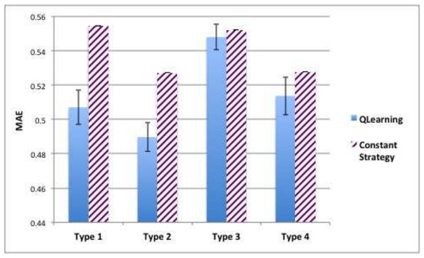


Fig. 2. Comparing ACPM performance with the best performing participant performance for each market type.

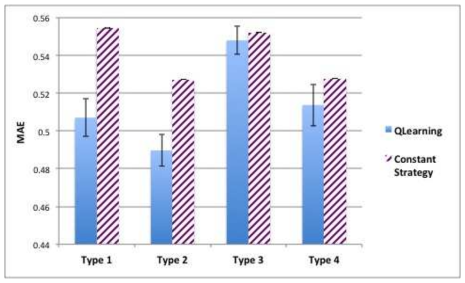


Fig. 3. Comparison of ACPM's performance with Q-learning and without.

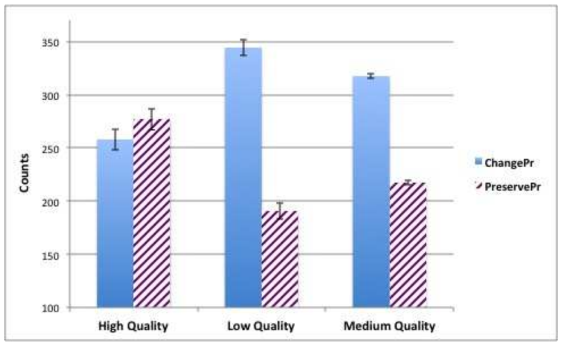


Fig. 4. Popularity of each action for agents accessing different quality of data streams.

two rounds and all participants use the Q-learning trading strategy. In these experiments, the market parameters, except C, are the same values as in the first set of experiments. Experiments indicated that, as the number of participating agents is relatively small, C is best set for each market to maximum error so that all agents receive positive rewards. Then the performance of ACPM is compared with same models mentioned above as benchmarks. They are run independently without the concept of ACPM. These models use same data as ACPM agents do, and similar to ACPM are run incrementally[10]. For each available record, they predict the true value and then are retrained again with the correct answer and all seen records. All models, both in ACPM and benchmarks, are implemented using Java Weka API (3-7-10) and configured with their default parameters.

Experiments In our first experiment (Figure 5) we compare ACPM's MAE with the MAE of each of the regression models and ensemble methods by means of the MAE of the agents that use the method as their analysis algorithm. We then go on in Figure 6 to

[10] Please note that their performance should not be compared with when they are run using batch training.

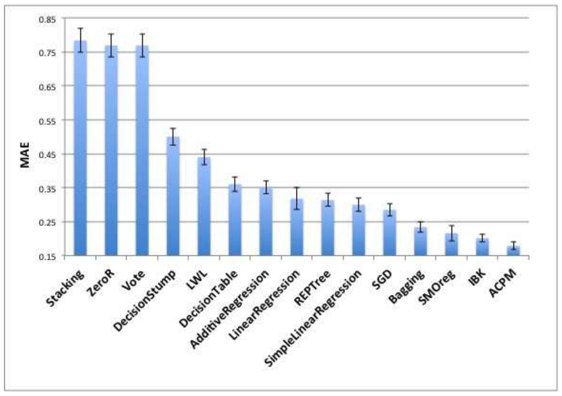


Fig. 5. Comparing ACPM performance with well known machine learning regression models and ensembles.

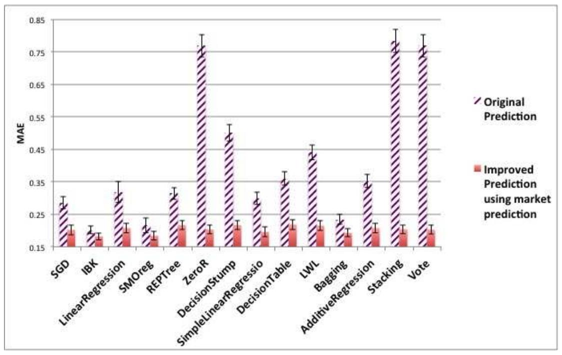


Fig. 6. How participating in ACPM and utilising Q-learning strategy improves the performance of each classifier.

compare the difference of MAE between classifier/ensemble if the agent was employing Q-learning or not.

Results Our experiments show (see Figure 5) that ACPM has a lower MAE than all regression models and ensembles (P-value is less than 0.001 for all except IBK (P-value= 0.08), SMOReg (P-value= 0.07)). They further demonstrate that an agent using well-known regression models can reduce its MAE when it uses Q-learning.

Figure 6 demonstrates that the performance of each classifier is improved by participating in the market and using the Q-leaning trading Strategy (highly significant for all except IBK and SMOReg).

3.4 Analysis

A number of our hypotheses are satisfied immediately from our experiments. Given that the MAE of ACPM is always lower than the best performing agent in any market type, we can safely state that it performs better (H1) and that the system is resilient to different proportions of low- and high-performing participants (H2). Based on our first experiment, it is not surprising that ACPM performs better than regression models or ensemble methods (H6), as demonstrated in Figure 5.

The system attains its high performance by granting more influence to those that have high quality data sources and effective analysis algorithms. The reward function rewards the market participants according to their prediction accuracy and the amount invested. Obviously, lower error and higher investment leads to higher revenue. In this way, agents are incentivised to make accurate predictions and adjust their investment based on their confidence in the prediction. After a few markets (records), the differences between agent capital becomes apparent as some of the agents gain revenue and some of the agents loose a proportion of their capital as a result of their performance. The integration function weights each prediction by the amount of investment. In this way, higher quality agents acquire greater influence in predicting the outcome of the event, since they gain more capital over time, and consequently can invest more in their bids.

The other reason for the performance of the system is that the agents learn to improve their prediction by considering market prediction as another source of information. Figures 3 and 6 show that Q-learning does improve each agent's performance and consequently the system's performance by adding a further reduction in prediction error; hence supporting hypotheses H3 and H7. Using the Q-learning trading strategy, each agent learns the extent to which it should use the market prediction to update its prediction. Therefore, while high quality agents ignore market predictions, low quality agents learn to minimise the amount of noise (low accurate prediction) they send to the market maker. This is demonstrated in Figure 4 and confirms H4 and H5. The validity of the ACPM approach through its application to several of the UCI data sets is confirmed, but cannot be presented here due to sake of space.

4 Related Work and Conclusion

We proposed an Artificial Continuous Prediction Market (ACPM) for predicting a continuous variable based on the integration of diverse data sources with different varying quality. It acts as an adaptive ensemble algorithm which is capable of shifting focus in response to changes in individuals' predictions.

To our knowledge, there is relatively little research on artificial prediction markets as a machine learning technique. Our work is different from related works in artificial prediction markets [12, 1, 15, 11, 8], prediction with expert advice and its subfields [16, 3, 4, 7, 10, 14, 9], opinion pools and all ensemble techniques as learning happens at two levels, i.e. market and agents. The market learns the weighting of each agent on the market prediction dynamically while participants revise their beliefs and can retrain themselves (i) after each round of a market by comparing their prediction with market prediction to maximise their utility in the current market. (ii) after each market in order to maximise their utility in future markets. Finally, we note that previous works are designed for discrete classification and our work is designed to predict a continuous variable.

Our next step is to develop an intelligent market that can self-select the appropriate parameters for the market based on the characteristics of market participants and their data sources. We also plan to apply ACPM on different domains, such as, for example, stock market and cancer predictions.

References

1. Adrian Barbu and Nathan Lay. An introduction to artificial prediction markets for classification. *The Journal of Machine Learning Research*, 13(1):2177–2204, 2012.
2. H. Chen, D. Zeng, P. Yan, and P. Yan. *Infectious Disease Informatics: Syndromic Surveillance for Public Health and Biodefense*. Integrated series in information systems. Springer Science + Business Media, 2010.
3. Yiling Chen and Jennifer Wortman Vaughan. A new understanding of prediction markets via no-regret learning. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 189–198. ACM, 2010.
4. Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
5. Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2008.
6. Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.
7. Elad Hazan. 10 the convex optimization approach to regret minimization. *Optimization for machine learning*, page 287, 2012.
8. Janyl Jumadinova and Prithviraj Dasgupta. Prediction market-based information aggregation for multi-sensor information processing. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, pages 75–89. Springer, 2013.
9. Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
10. Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.
11. Jono Millin, Krzysztof Geras, and Amos J Storkey. Isoelastic agents and wealth updates in machine learning markets. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1815–1822, 2012.
12. Johan Perols, Kaushal Chari, and Manish Agrawal. Information market-based decision fusion. *Management Science*, 55(5):827–842, 2009.
13. Russ Ray. Prediction markets and the financial "wisdom of crowds". *Journal of Behavioral Finance*, 7(1):2–4, 2006.
14. Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.
15. Amos J. Storkey. Machine learning markets. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudk, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 716–724. JMLR.org, 2011.
16. Vladimir G Vovk. A game of prediction with expert advice. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 51–60. ACM, 1995.
17. Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

A Multi-agent Auction-based Approach for Modeling of Signalized Intersections

Mehdi Mashayekhi[1], George List[1]

[1]North Carolina State University, Raleigh, North Carolina, USA
{mmashay2, gflist}@ncsu.edu

Abstract. This paper shows how the traffic interactions of an intersection can be regulated by means of auction theory, multi-agent systems and machine learning techniques. In the proposed work, the intersection space is viewed as a spatially complex set of scarce commodities whose temporal non-overlapping usage is adjudicated through an interactive bidding process that involves multiple agents. A machine learning technique is used to optimize the bidding strategies of the agents. The multi-tiered decentralized agent-based framework proposed here increases modularity by decomposing the intersection into smaller sub-parts, adds flexibility and effectiveness by designing it to account for variable stage sequence. Moreover, unlike most other control strategies which are either based on delay or queue length, the proposed method is based on both of these traffic-related metrics. The proposed method is used to regulate the traffic for a network of six intersections and its results are compared with two other control strategies including pre-timed and fully actuated.

Keywords: Multi Agent Systems, Machine Learning, Auctions, Adaptation, Traffic Signal

1 Introduction

The efficient and effective movement of people and goods is critical if society is to achieve economic prosperity, energy efficiency, environmental sustainability, global competitiveness and other objectives. Much of the transportation activity that occurs in urban areas is by highway, and much of that is via surface arterials.

The single biggest impediment to efficient operation of surface arterials is signal timing. While traffic signals are necessary to ensure the safe movement of all vehicles, their use reduces efficiency. Trips take longer than they would if the vehicle-to-vehicle conflicts could be eliminated without signal control. Thus, optimizing the signal control and making it traffic adaptive is critically important.

Off-line optimization has been standard practice for a long time. Today there are well-established algorithms such as TRANSYT [1], which generate optimal coordinated plans for fixed-time operation. The main weakness of such methods is that their plans are computed for a static situation, based on historical data. But that situation never actually exists in the network. Other systems, like SCOOT, Split Cycle and

Offset Optimization Technique [2], are similar to TRANSYT but are traffic-responsive (they use real-time data from detectors to update the signal control settings). SCATS, the Sydney Coordinated Adaptive Traffic System [3], also makes real-time decisions about the control strategy based on detector inputs. The main difference between SCOOT and SCATS is that the latter is a hierarchical system while the former is not.

The strength of systems like SCOOT and SCATS is their ability to adapt to the traffic flow conditions. However, these systems have limitations. They work with a fixed cycle length. Their coordination patterns have to repeat on a single-cycle basis, and their responsiveness is slow (tempered) to ensure stable operation. Their off-line or even on-line optimization of the signal timings has difficulty adjusting quickly to changing traffic patterns. And yet, traffic flows are highly dynamic. Thus, optimal signal timing plans are difficult to determine in advance.

PRODYN [4], OPAC [5], and UTOPIA [6] are also examples of adaptive systems that are not centralized, but their relatively complex computation and communication schemes make their deployment costly [7].

Given the growth rates foreseen for urban traffic in the future, more flexible and robust approaches are necessary. Hence, making signals smarter is the objective of this paper. The intent is to make signal control more sensitive to the evolving traffic streams and more intelligent about coordination. This will produce savings in energy consumption, pollutant emissions, and delays.

During the last few years, multi-agent systems (MAS) have become a promising application domain within artificial intelligence (AI) for optimizing traffic signal control. MAS techniques can be applied to situations where the conditions evolve dynamically. They can capture the important details at the level of individual entities and produce useful control results. They can be used in a variety of ways to emulate system behavior. They can be active, heterogeneous participants in an environment representing the system of interest and engage in information processing and decision making. Their behavior can be visualized, monitored, and validated at individual agent level, leading to new possibilities for analyzing, debugging, and developing signal control strategies.

This paper shows how the traffic interactions of an intersection can be regulated by means of auction theory, MAS, and machine learning (ML) techniques. In the proposed work, the intersection space is viewed as a spatially complex set of scarce commodities whose temporal non-overlapping usage is to be adjudicated through an interactive bidding process that involves multiple agents. A ML technique is used to optimize the bidding strategies of the agents. The proposed method offers the following features and characteristics: 1) decentralized design and operation, which is usually less expensive comparing to centralized approaches; 2) variable staging sequence, i.e., it is not hampered or constrained by a prescribed stage sequence as is common with all actuated, semi-actuated coordinated, and pre-timed control strategies which have fixed staging sequence; 3) control logic based simultaneously on queue length and delay (unlike most other control strategies which are either based on delay or queue length); 4) self-learning, i.e., decreases human intervention in the operation after implementation; 5) model-free, i.e., does not need a model of traffic pattern that is challenging to acquire; 6) robust, i.e. with no single point of failure.

The paper is organized as follows: Section 2 reviews some related research. Thereafter, Section 3 presents the details of the proposed model. Section 4 presents the experimental results that have been carried out in a network of six intersections. Finally, Section 5 gives some conclusions and points out lines of future work.

2 Related Work

MAS involve using distributed intelligence, often autonomous, to develop problem solutions. For example Choy et al. [8] present a hierarchical MAS that consist of three layers of agents: controller agents, zone controller agents, and regional controller agents. The implementation of agents is based on feed-forward neural network and fuzzy logic theories.

MAS often use ML to adapt to the evolving traffic conditions. For example, Steingrover et al. [9], Weiring [10] employ a reinforcement learning technique to minimize the overall waiting time of the vehicles. Here the learning task is represented as a feedback loop focused on the aggregated waiting times for individual vehicles. Another example is the work done by Tantawy et al. [11] in which a MAS is proposed for adaptive traffic signal control. In their proposed approach each agent (which controls one intersection) plays a game with its immediate neighbors and learns and converges to a response policy to all neighbors' policies using a reinforcement learning technique.

Some researchers suggested adapting market-based ideas to traffic signal control. Isukapti and List [12] have demonstrated that auction theory can be used as a paradigm for modeling signal control. Vasirani and Ossowski have also demonstrated that a distributed, market-inspired, mechanism can be developed for the management of an urban road network, where drivers trade with the infrastructure agents in a virtual marketplace, purchasing reservations to cross intersections [13]. Carlino et.al [14] have shown that auctions can be run at each intersection to determine the order in which drivers perform conflicting movements. In their approach autonomous vehicles (which are considered as agents) bid on behalf of the travelers. This approach has been used for an isolated intersection. But how the agents (drivers) bid is not clear and well explained. More importantly no strategy for optimizing the bids is proposed or implemented.

A comprehensive literature review of agent-based technology for transportation systems can be found in [15].

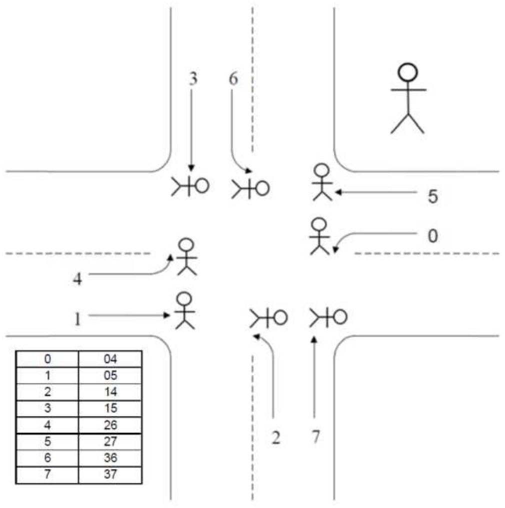
3 Methodological Approach

Following the lead of Isukapati and List [12], the intersection space is viewed as a spatially complex set of scarce commodities whose temporal non-overlapping use is adjudicated through an interactive bidding process that involves multiple agents. To further ensure safety, change intervals are included (i.e., a yellow interval followed by an all red) and minimum greens are used. No maximum greens are employed. FIFO-based discharge strategy is employed to ensure that realistic separations are maintained between vehicles as they are discharged from the stopbar.

Each movement is managed by an agent called "movement manager". In a typical intersection this means there are eight movement managers operating in parallel as shown in Fig. 1. There is another agent called "intersection manager" that control the behavior of movement managers.

Bids occur each time the use of the intersection is contested. This arises when agents for conflicting movements have non-zero queues and transitioning to a new stage is possible. Transitions are possible at the end of the minimum green for the current stage or at the end of every ensuing discharge headway or at the end of each maximum gap time if there is no discharge headway.

Stages (phases) are formed by combinations of winning bidders. Eight stages (movement combinations) are possible for the intersection shown in Fig. 1. The stages are shown in the table in the left-hand bottom corner of the Fig. 1. First column shows the stages, and the second column shows the movement combinations. For example, stage 3 is formed by combining movement 1 and 5.



0	04
1	05
2	14
3	15
4	26
5	27
6	36
7	37

Fig. 1. Structure of the agent based modeling approach

The bidding process is as follows. Each movement manager (with a non-zero queue) submits a bid and the intersection manager sums these bids for all possible, safe, movement combinations (eight in this case). The pair of movements with the highest combined bid win. For example, if the bids are 1, 6, 2, 5, 3, 2, 4, 5 respectively for movements 0, 1, 2, 3, 4, 5, 6, 7, the combined bids would be 4 (04), 3 (05), 9 (14), 8 (15), 6 (26), 7 (27), 9 (36), 10 (37). Movements 3 and 7 would win and pays the intersection manager what was bid (first-price bidding).

To minimize the vehicle to infrastructure (V2I) dependence, consistent with the current state of the practice in rich intelligent transportation systems (ITS), movement managers receive tokens when vehicles join their queues. They use these tokens as the basis for submitting their bids and they "pay" tokens to the intersection manager when they win bids. In the simulation-based realizations presented here, the movement managers have access to information about their respective movements and limited information about the other movement managers. They know how many drivers are in queue, how many tokens they have, the bid that they submit, and what the winning bid was.

As indicated, when movement managers win following one or more prior losses, they receive use of the intersection for a minimum green. When the minimum green expires, they have to bid again to retain control at the end of each discharge headway (assuming they have a queue) or minimum gap time to see if they can continue to retain control of the green. When bids are to be submitted, those movement managers with non-zero queues are allowed to do so. Those without queues are excluded. If the movement managers currently holding control of the intersection submit winning bids, they are allowed to continue discharging vehicles for one more discharge headway or the minimum gap time (i.e., 3 seconds), whichever is smaller. If they lose control, then, then a clearance interval is imposed and control shifts to the new winners for a minimum green.

Reinforcement learning has been used to help the movement managers improve their bidding strategy based only on the knowledge of their own past received payoffs. Specifically, a Q-learning approach is used to create an optimal bidding strategy for each movement manager.

The core of the Q-learning algorithm is a Q-table and an algorithm for updating the Q-table and choosing actions. A Q-table *Q(s, a)* is a matrix indexed by state *s* and action *a*, which is the expected discounted reinforcement of taking action *a* in state *s*. At each time, an agent is assumed to be in a certain state *s*, and it chooses an action *a* according to the Q-table and other algorithms to interact with the environment. Then the agent receives a reward *r* from performing action *a* and observes a new state *s'*. After that, the Q-table is updated by the following equation:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(\mathrm{r} + \gamma \max_{a'}(\mathrm{s}', \mathrm{a}')) \qquad (1)$$

Where α is the learning rate, and γ is the discounting factor. Q-Learning can be summarized in following steps:

1. Let the current state be *s*
2. Choose an action *a* to perform
3. Receive a reward *r* from performing action *a* and the resulting state *s'*.
4. Update *Q(s,a)* using equation (1).
5. Go to step 1).

The design elements of the Q-learning in terms of the typical structure of it for each movement agent are:

- *State*: The current queue
- *Action*: the bid amount to submit

- *Reward*: the immediate reward is composed of two terms. The first term is a function of the delay and is defined as the change (saving) in the total cumulative delay, i.e., the difference between the total cumulative delays of two successive decision points (bidding cycles). The second term is a function of winning the bid. It is zero if the movement agent loses the bid. On the other hand, if the movement agent wins the bid, the reward is based on the difference between the number of vehicles discharged and the submitted bid. For example, if the movement agent wins the bid and receives a minimum green (e.g., 6 seconds), the average headway is 2 seconds, and the submitted bid was 1.5 tokens, then the movement agent is able to discharge 3 vehicles (6/2) and the reward would be 1.5 (3-1.5).

Movement managers use an ϵ-greedy selection approach in order to explore the state and action environment. That is, random actions are selected with probability ϵ and the actions with the highest Q-values are chosen with a probability of (1-ϵ). The exponential function (e^{-En}) is used for ϵ-greedy approach in which E is a constant and n is the number of iteration. The exponential function is a simple ϵ-greedy exploration approach with a gradually decreasing rate of exploration. By using this approach, at the starting the agent mainly explores, as it has no previous information to exploit, and the agent moderately increases the degree of exploitation towards the end of the learning process. Gradual shifting is necessary to ensure that the entire state space is covered during the learning process. As suggested in the literature [16] the learning rate is considered as follows:

$$\alpha^k = \frac{E}{v^k(s,a)} \tag{2}$$

Where α^k is the learning rate at time *k*, *E* is a constant, and $v^k(s, a)$ is the number of visits to a particular state-action pair *(s, a)*. This learning rate allows to start with high learning rate at first to allow for fast modifications then use lower rates as time progresses.

The overall pseudo code for the model is summarized in Fig. 2.

```
Input:
   - $\delta t$ =  0.1 (initialize time step)
   - $t$ =  0 (simulation clock)
   - $g_{min}$ = 6.0 (minimum green)
   - $c$ = 4.0 (change interval, i.e., yellow and all red intervals)
   - $h_d$= 2.0 (headway)
   - $f_{\min}$ = 1.0 token (initial fee for each vehicle)
Initialize: for each movement manager initialize Q values as zeros
The algorithm:
  while $t < T$ (simulation time period):
    $t = t + \delta t$
    for each movement manager
        if  <new arrivals = True>
          collect  $f_{\min}$
          add new arrivals to service queue
    if the use of intersection is contested
       for each movement manager who submitted bid in previous bidding cycle
           compute the total cumulative delay
           compute the reward
           update the Q values
       for each movement manager with non-zero queues
           if explore ($\epsilon$) then
              submit random bid (action) for the current queue (state)
              (bid is subject to { $bid_{\min} = 0.5 * \frac{g_{\min}}{h_d}$ , $bid_{\max} = \frac{bankBalance}{queueLength} * \frac{g_{\min}}{h_d}$ })
           else
              submit bid according to the policy
       intersection manager sums submitted bids for all possible, safe, movement combinations (eight)
       the pair of movements with the highest combined bid will be selected
       wining movement managers pays the intersection agent amount equal to winning bid
       if < current winner = previous winner >
           intersection manager extend green by 3 seconds
       else:
           intersection manager impose a clearance time of 4 seconds
           intersection manager allocate intersection control to new winners for $t = t + g_{min}$
```

Fig. 2. Pseudo Code for control structure of simulation

4 Case Study

Fig. 2 shows the network of intersections which has been modeled (each intersection simply applies the control strategy presented in the previous section). The east-west arterial has four intersections (0, 1, 2, and 3) while the north-south arterial has three (4, 2, and 5). Intersection 2 is common between them. Left turn bays exist everywhere. The figure also shows the travel distances between the intersections. The longest distance is 800 feet and the shortest is 550 feet with the nominal travel speed at 30 mph.

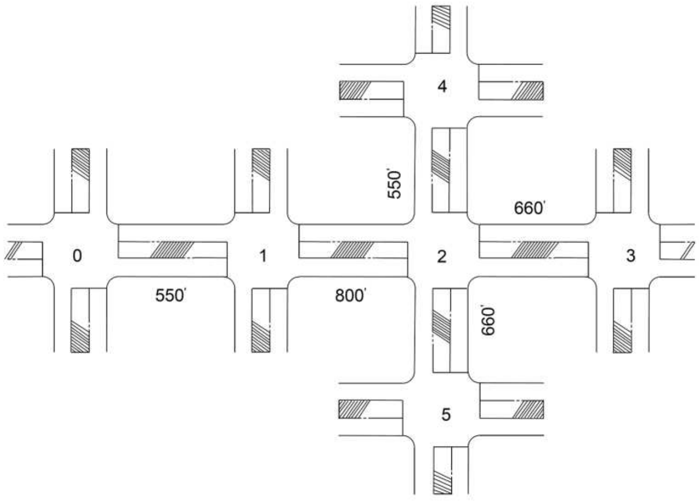


Fig. 3. Modeled network of intersections

Two different combinations of traffic volume shown in Table 1 are considered in this paper. Case 1 represents a "nominal" or baseline condition for the network. The heavier flows are east-west, with 600 vehicles per hour (vph) entering the network in these two directions. Side street volumes along the east-west arterial are 200 vph. The north-south arterial has slightly less traffic, with main flows at 400vph, both north and southbound. The side street volumes are 150 vph. In all cases, for simplicity, the turning percentages are 15% left turns and 10% right turns.

The other case makes adjustments to the baseline values. All of the volumes increase by 25%. Thus, the street volumes grow to 750 vph east-west and 500 vph north-south.

Table 1. Volume combination for each intersection.

	Intersection					
Approach	0	1	2	3	4	5
			Case 1			
North	200	200	-	200	-	**400**
South	200	200	-	200	**400**	-
East	**600**	-	-	-	150	150
West	-	-	-	**600**	150	150
			Case 2			
North	250	250	-	250	-	**500**
South	250	250	-	250	**500**	-
East	**750**	-	-	-	188	188
West	-	-	-	**750**	188	188

Three signal control strategies are explored. These are: pre-timed, fully actuated, and the proposed control strategy. The three of these control strategies are implemented in the custom-built traffic simulator developed by the authors.

For the pre-timed control strategy, Synchro7 [17] which is an analysis and optimization program for optimizing signal timing for arterials, was used to develop optimal signal timings. The obtained signal timing plans including cycle lengths, splits, phase sequences and offsets for each intersection and in each case were implemented in the custom-built traffic simulator.

For fully actuated control strategy, the implemented logic is as follows. Normal stage rotation is followed including the feature that in the absence of any requests for service, the controller returns to the main street stage. This means the user needs to specify a main street stage and minimum and maximum greens for each movement. In the implemented logic here, for intersections 0, 1, 2, and 3, stage 3 is considered the main street stage, and for intersections 4 and 5, stage 7 is considered as the main street stage. Minimum green is 5 seconds and maximum green is 40 seconds.

Fig. 3 and Fig. 4 show the results of the simulation for two traffic volume cases. As can be seen from the figures, the proposed control strategy improves as time passes which make sense. Initially the proposed control strategy does a lot of exploration and as the time passes it learns the best control policy and does more exploitation.

In case 1 (Fig. 3) which represents a low traffic volume, pre-timed control strategy is the poorest and the proposed control strategy is the best and actuated control strategy performs in the middle. Again, the results make sense, imposing a pre-defined control strategy (pre-timed) when the traffic volume is low is not a good strategy.

In case 2 (Fig. 4) which represent a heavy volume of traffic, pre-timed control strategy is performing better than the other two control strategies, the proposed strategy is the second and fully actuated has the poorest performance. Again, the results make sense since when there are heavy traffic, it is better to impose some pre-defined timing plan to regulate the traffic.

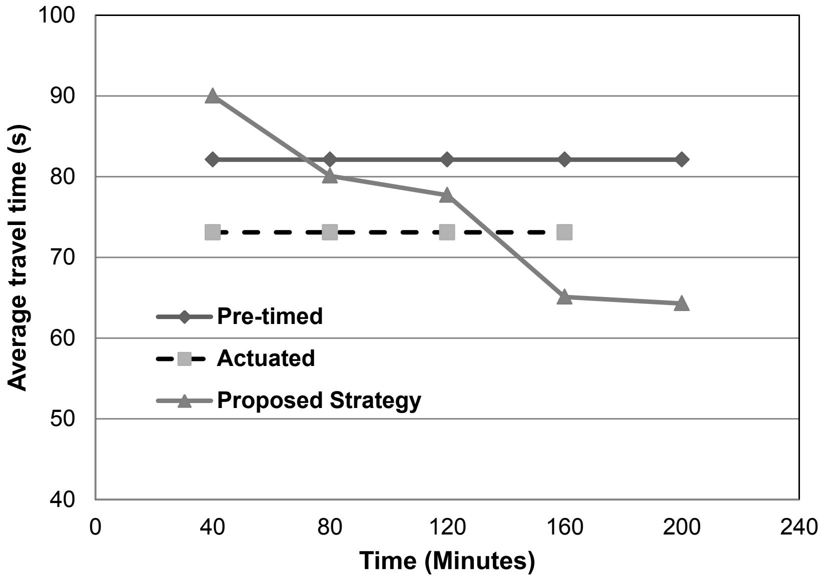


Fig. 4. Results of case 1

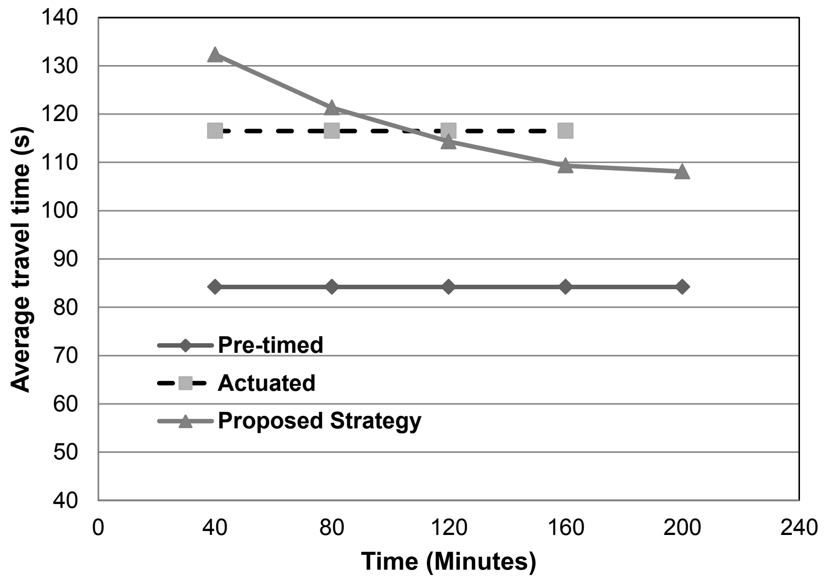


Fig. 5. Results of case 2

5 Conclusion and Recommendation

In this paper a new control strategy is presented which integrates MAS, ML technique, and auction theory to regulate the movement of cars at an intersection. This multi-tiered agent-based framework provides a way to decompose the intersection into smaller sub-parts. It increases the efficiency of the model, provide a way to create a parallel computing realization, and enhance scalability.

The authors believe that this area of research is one that has significant promise for the future, especially in light of the increasing demands for more features and capabilities, especially real-time control, being placed on advanced traffic management systems.

There are various venues for further research. Our own future research will concentrate on:

- applying other exploration functions other than ϵ-greedy such as softmax and ϵ-softmax
- second-price bidding is another area of future research. In this bidding strategy, truthful bidding is a dominant strategy. The fact that truthfulness is a dominant strategy also makes second-price auctions conceptually very attractive.
- in the current model, the intersection managers are not cooperating, so the other area of future research is to make intersection mangers cooperative and see how it affects the performance of the whole network.
- linkage of the developed signal control logic to a commercially available traffic simulator (e.g., VISSIM [18], Paramics [19]) to test the performance of the developed control logic for traffic scenarios similar to real world cases and compare its results with other adaptive controller such as SCOOT and SCATS.

References

1. Robertson, D.: TRANSYT method for area traffic control. In: Traffic Eng. Control, vol. 10, pp. 276–281 (1969)
2. Robertson, R., Bretherton, R. D.: Optimizing networks of traffic signals in real time - the SCOOT method. In: IEEE Trans. Veh. Technol., Vol. 40, pp. 11-15 (1991)
3. Sims, A.G., Dobinson, K.W.: The sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. In: IEEE Trans. On Vehicular Technology, 29(2), pp. 130-137 (1980)
4. Henry, J., Farges, J. L., Tuffal, J.: The PRODYN real time traffic algorithm. In Proceedings of the International Federation of Automatic Control (IFAC) Conference, Baden-Baden: IFAC (1983)
5. Gartner, N. H.: OPAC—A demand-responsive strategy for traffic signal control. Transportation Research Record, 906, pp. 75–81 (1983)
6. Di Taranto, M.: UTOPIA. In Proceedings of the IFAC-IFIP-IFORS Conference on Control, Computers, Communication in Transportation, Paris. ifac (1989)
7. Bazzan, A.L.C.: Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Autonomous Agents Multi-Agent Syst., vol. 18, no. 3, pp. 342–375, Jun (2009)

8. Choy, M.C., Srinivasan, D., Cheu, R. L.: Cooperative, hybrid agent architecture for real-time traffic signal control. In: Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 33.5, pp. 597-607 (2003)
9. Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., Bakker, B.: Reinforcement Learning of Traffic Light Controllers Adapting to Traffic Congestion. In: BNAIC (2005)
10. Wiering, M.A., Veenen, J., Vreeken, J., Koopman, A.: Intelligent traffic light control." Institute of Information and Computing Sciences. Utrecht University (2004)
11. El-Tantawy, S., Abdulhai, B., Abdelgawad, H.: Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and largescale application on downtown toronto. In: Intelligent Transportation Systems, IEEE Transactions on 14(3), pp. 1140–1150 (2013)
12. Isukapati, I.K., List, G. F.: Agent Based Framework for Modeling Operations at Isolated Signalized Intersections. In: Elsevier, Procedia - Social and Behavioral Sciences (2013)
13. Vasirani, M., Ossowski, S.: A computational market for distributed control of urban road traffic systems. In: IEEE Transactions on Intelligent Transportation Systems 12(2), pp. 313–321 (2011)
14. Carlino, D., Boyles, S.D., Stone, P.: Auction-based autonomous intersection management. In: IEEE ITSC'13 (2013)
15. Bazzan, A.L.C., KLUGL, F.: A review on agent-based technology for traffic and transportation. In: The Knowledge Engineering Review, vol. 29:3 (2013)
16. Gosavi, A.: Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. In: Springer, Netherlands (2003)
17. Husch, D., Albeck, J.: Trafficware SYNCHRO 7 User Guide, Trafficware, Albany, Calif, USA (2006)
18. PTV Plannung Transport Verkher AG.: PTV, VISSIM 5.40-03 User Manual, Karlsruhe, (2012)
19. Quadstone Paramics.: Paramics Microscopic Traffic Simulation Software, http://www.paramics-online.com

Multi-Objective Reinforcement Learning through Reward Weighting

Felipe Leno da Silva[1] and Anna Helena Reali Costa[1]

Intelligent Techniquesq Laboratory (LTI)
Escola Politécnica - Universidade de São Paulo, São Paulo, Brazil.
`f.leno@usp.br, anna.reali@usp.br`

Abstract Complex, real-world domains such as Smart Grid scenarios require the optimization of multiple, and often conflicting, objectives. Several works proposed solutions in this area using Distributed W-Learning (DWL), a multi-agent and multi-objective algorithm able to deliver a good performance. However, these works rely on an assumption that the rewards for all objectives can be extracted in the same temporal frequency. When the reward functions have different time frequencies, DWL prioritizes the objective that gives the most frequent feedback, which is not always the most important objective. In order to solve this problem, we here propose the Single Policy Distributed W-Learning (SP-DWL) algorithm, an extension of the DWL algorithm that relies on a reward weighting function to deal with multiple objectives, without losing the ability of agent cooperation. Our experiments show that SP-DWL is able to balance conflicting objectives even when reward functions are defined in different time frequencies.

Keywords: Reinforcement Learning, Multi-Agent Systems, Smart Grid, Distributed W-Learning

1 Introduction

Modern computing systems try to solve problems with an ever increasing complexity. In past years, Machine Learning solved many complex problems for which the traditional programming approach was unable to provide feasible solutions. Reinforcement Learning (RL) [10] is an area of Machine Learning in which an agent observes the environment, chooses and executes an action to affect the environment and to see how good that action was. RL has been successfully applied in many domains, since it does not require much built-in domain knowledge and achieves good results after the agent training.

However, in complex domains (where there are too many states to be considered), learning the optimal behavior of an agent can be too slow or too difficult

[1] The authors are grateful to CNPq (grant 311608/2014-0) and to Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE) for the financial support.

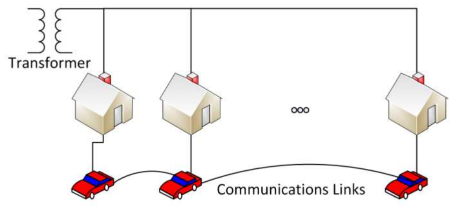


Figure 1: The Smart Grid scenario used in this paper (image modified from [11]). There are a certain number of houses in a neighborhood, each with an electric vehicle (EV). Each EV has a communication link with the next and the previous car, and EVs need to coordinate their charging to prevent the transformer overload.

for many applications, in which RL with a single agent becomes computationally expensive. For this kind of problems, multiple agents acting in one system can share workloads, be robust to individual failures, and scale well with the addition of extra agents [14]. Thus, Multi-Agent Systems (MAS) can benefit from cooperation between agents, and how this cooperation will occur depends on how the agents can exchange information [8].

One of such real world complex domains is the Smart Grid, for which some multi-agent approaches have been proposed in various scenarios [4,5,11,12,13].The scenario here considered consists in a neighborhood of houses connected to a transformer, and each of the houses has an Electric Vehicle (EV), as illustrated in Figure 1. Each EV must charge its battery to allow the user's daily travel, while avoiding the transformer overload (which happens when a large number of EVs is charging at the same time).

This problem can be modeled as a Multi-Objective System, in which each EV tries to optimize two conflicting objectives: Charging the battery; and avoiding the transformer overload. Several works have been published with proposals for this scenario [5,11,12]. All these works used Distributed W-Learning (DWL)[4] as the basis for their proposals. However, one problem that was not addressed in any of these works is that, in the reward scheme published in [5], every EV knows beforehand how much battery is needed to successfully accomplish its daily journey. However, this is a strong assumption, since it is usually not possible to know exactly how much battery is needed before the user tries to use it: The battery consumption can change due to traffic conditions or to unpredictable route and destination changes, for example. A more realistic reward function would evaluate if the battery was sufficient only after the user performs his daily journey. However, this approach would result in reward functions that give feedbacks on different temporal frequencies. The transformer load can be verified in a window of minutes, whereas the battery policy can give just one feedback per day (after the journey).

In this situation, the DWL ability to balancing the objectives is hampered, and the goal that provides the most frequent feedback is prioritized over the other. Since for many domains the goal with higher temporal frequency is not the most important objective, the agent actuation may be unsatisfactory. In order to solve this problem, we propose a modification in DWL, here called Single-Policy Distributed W-Learning (SP-DWL), where all objectives are solved through a single policy and the multi-objectives are weighted through Reward Weighting. In our framework, "weighting" rewards consist in applying a weighting function to encapsulate multiple reward signals as: $R = F(\boldsymbol{r})$, where R is the resulting reward function, F is a reward weighting function, and $\boldsymbol{r}$ is a vector of multiple reward signals. A well-designed Reward Weighting function leads DWL to prioritize a desired objective, even if its feedback's temporal frequency is not the higher one.

This paper is organized as follows: Section 2 provides a review of relevant literature on RL, MAS, Multi-Objective Systems and DWL; Section 3 details our proposal, outlining its strengths and weaknesses; Section 4 describes the modeling of the Smart Grid problem we are trying to solve; Section 5 presents our experiments and their results, along with discussions; Finally, Section 6 concludes the paper and points toward future works.

2 Background

In the basic pattern underlying Reinforcement Learning (RL), an agent has a set of sensors to observe the state of the environment and a set of actions through which it can act in the environment, alter the current state, and see how good that action was [10]. This process can be repeated multiple times, and after a number of learning steps the agent will have learned how to act optimally given the current state. RL usually uses a Markov Decision Process (MDP) to describe the problem and environment. In an MDP, a combination of variables that describe the environment defines a discrete state and the MDP contains all possible states. Each state has a set of applicable actions, and each action causes a state transition.

The agent can assess the quality of its actions through a reward signal provided by a supervisor or the environment, and this reward will be used to update a Value Function that is stored by the agent and represents how good a particular state is. When the Value Function converges for all states (i.e., its values no longer change after updates), the agent will have learned how to behave optimally for each possible state for a given objective (encoded by the reward function). The knowledge of how to behave given a state is represented by a policy, which is a representation of the best action for each possible state. An RL algorithm that iteratively learns an optimal actuation policy with basis on interactions of the agent with the environment is the Q-Learning algorithm [10], where the Q table is updated as: $Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha[r_i + \gamma \underset{a}{max} Q(s_{t+1}, a)]$, where r_i is the received reward after executing action a_t on state s_t, α is the learning rate ($0 < \alpha \leq 1$), and γ is the discount factor. After learning, the optimal policy is given by: $\pi^*(s) = \underset{a}{argmax}\ Q(s, a)$.

2.1 Multi-Agent and Multi-Objective Systems

In Multi-Agent Systems (MAS), there is more than one agent acting in the environment at the same time, which means that the state transition no longer depends solely on an agent's action, but on the combination of all actions. Since, for most domains, it is infeasible to learn a policy considering the product of each state and each action for all agents, agents may try to learn a joint policy representative of a Nash equilibrium (strategy where no agent would benefit from changing their own policy assuming all agents will stick to their current policies) or a Pareto optimal joint policy (there is no other policy that can achieve a better reward for an agent without reducing another agent's reward) [3]. There are many approaches for MAS, using the RL framework or not, that are not regarded in this paper. For more comprehensive reviews see [1,2].

Many complex domains require the ability of handling multiple objectives. A possible approach is to define a reward for each objective [9], however, dealing with multiple objectives remains a difficult task, in which each objective may require conflicting actions. The domain addressed in this paper has conflicting objectives, since one objective is to charge the battery of all cars, while the other is to avoid the transformer overload. One trivial solution for the first objective is to always charge all cars until the battery is full; however, this policy is likely to overload the transformer at all time steps. The trivial solution for the second objective is to never charge any car, which will assure no transformer overload, but will charge no car. Therefore, a good policy for our problem must balance between these two objectives.

2.2 Distributed W-Learning

Distributed W-Learning (DWL) [4] is a multi-agent and multi-objective RL algorithm. In DWL, each agent has a communication channel with a certain number of neighbor agents, with which this agent can communicate and collaborate. The agent has a set of Local Policies LP_i and a set of Remote Policies RP_i. Each Local Policy π_l deals with a single objective and ignores all other agents and objectives, learning only the local effect of each action. In turn, a Remote Policy π_k encodes how a particular action affects a neighbor's reward; each agent will thus learn how to collaborate with its neighbors through Remote Policies. Each agent has one Remote Policy for each objective of each neighbor (see Figure 2).

As shown in Algorithm 1, at each time step, each policy (remote and local) suggests its optimal action, and the best action for a given time step is chosen by finding the maximum W-value associated to each policy. The W-Value represents how important for state s the policy π_i is at a given time step, and it is updated only when the suggestion of this policy is not obeyed [4]:

$$W_{\pi_i}(s) = (1-\alpha)W_{\pi_i}(s) + \alpha(Q_{\pi_i}(s, a_{\pi_{win}}) - (r_{\pi_i} + \gamma \underset{a}{max} Q_{\pi_i}(s', a))), \quad (1)$$

where $W_{\pi_i}(s)$ is the W-value of the current state s for policy π_i, $Q_{\pi_i}(s, a)$ is the Q-value of the state-action suggested by policy π_i, $a_{\pi_{win}}$ is the action suggested

Algorithm 1 DWL at each learning step

```
 1: for each learning step $t$ do
 2:     for each $Ag_i$ in $Ag$ do
 3:         //Get action suggestion by Local Policies (LP)
 4:         for each $\pi_l$ in $LP_i$ do
 5:             Observe $\pi_l$'s current state $s_l$.
 6:             Suggest action $a_l$ with max $Q_{\pi_l}$ for $s_l$
 7:         end for
 8:         $W_{LP} = \underset{\pi_i \in LP_i}{max} W_{\pi_i}$
 9:         //Get action suggestion by Remote Policies (RP)
10:         for each $RP_{i,j}$ in $RP_i$ do
11:             for each $\pi_k$ in $RP_{i,j}$ do
12:                 Get $\pi_k$'s state $s_{j,k}$ from $Ag_j$
13:                 Suggest action $a_{j,k}$ with max $Q_{\pi_k}$ for $s_{j,k}$
14:             end for
15:         end for
16:         $W_{RP} = \underset{RP_{i,j} \in RP_i}{max} \; \underset{\pi_i \in RP_{i,j}}{max} W_{\pi_i}$
17:         $W_{win} = max(W_{LP}, C \times W_{RP})$
18:         Execute winning action $a_{win}$ associated to $W_{win}$
19:         //Update policy values for Local Policies
20:         updateLPValues($W_{win}$,$a_{win}$,$s_l$)
21:         //Update policy values for Remote Policies
22:         updateRPValues($W_{win}$,$a_{win}$,$s_l$)
23:     end for
24: end for
25: procedure updateLPValues($W_{win}$,$a_{win}$,$s_l$)
26:     for each $\pi_l$ in $LP_i$ do
27:         Observe $\pi_l$'s current state $s'_l$
28:         Get reward $r_{\pi_l}$ from environment
29:         Update $Q_{\pi_l}$ using $(s_l, a_{win}, r_{\pi_l}, s'_l)$
30:         if $W_{\pi_l} \neq W_{win}$ then
31:             Update $W_{\pi_l}(s_l)$ (Equation 1)
32:         end if
33:     end for
34: end procedure
35: procedure updateRPValues($W_{win}$,$a_{win}$,$s_l$)
36:     for each $RP_{i,j}$ in $RP_i$ do
37:         for each $\pi_k$ in $RP_{i,j}$ do
38:             Get $\pi_k$'s state $s'_{j,k}$ from $Ag_j$
39:             Get reward $r_{j,\pi_k}$ from $Ag_j$
40:             Update $Q_{\pi_k}$ using $(s_{j,k}, a_{win}, r_{j,\pi_k}, s'_{j,k})$
41:             if $W_{\pi_k} \neq W_{win}$ then
42:                 Update $W_{\pi_k}(s_{j,k})$ (Equation 1)
43:             end if
44:         end for
45:     end for
46: end procedure
```

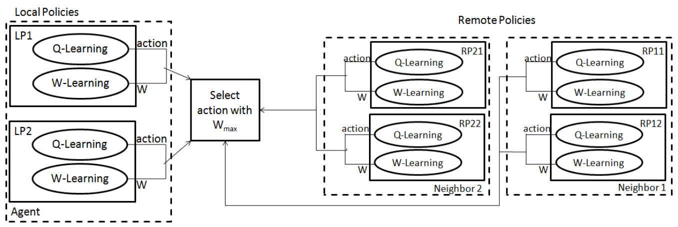


Figure 2: DWL action selection for an agent with two neighbors and two objectives. Dashed rectangles separate which policy refers to which agent (local agent or neighbor). There are a total of 6 policies in this scenario.

by the winning policy π_{win}, $Q_{\pi_i}(s', a)$ is the Q-value associated with the best action for the next state s', r_{π_i} is the reward received, α is the learning rate, and γ is the discount factor.

W-values are associated to states, rather than to state-action pairs. Q values are updated in every decision step as in the Q-learning algorithm (Section 2). The W-value for Remote Policies is weighted by a cooperation parameter, thus, the winning policy at each time step is chosen according to: $W_{win} = max(W_{LP}, C \times W_{RP})$, where W_{LP} is the highest W-value for all Local Policies, W_{RP} is the highest W-value for all Remote Policies, and C, $0 \leq C \leq 1$, is a cooperation parameter. Note that, in order to update each Remote Policy at each decision step, a communication with each neighbor will be necessary, from which the agent will receive its neighbor's current state and reward. An additional communication will be needed at the beginning of the learning process, in which each agent informs all neighbors its state space for each Local Policy.

3 Single Policy Distributed W-Learning

Here we describe our contribution, the Single Policy Distributed W-Learning (SP-DWL) algorithm, which is a variation of DWL [4] that deals with only one Local Policy per agent, while maintaining the cooperation ability. SP-DWL operates in a state space containing all variables of all objectives, while DWL has several policies that operate in a subset of state variables that matter for a given objective and arbitrates finding the most important objective for a given moment. A system implementing SP-DWL is composed of the following elements: (i) A set of agents, $Ag = \{Ag_1, \ldots, Ag_Z\}$, where Z is the total number of agents in the system; (ii) A set of neighbors for every agent, $N_i = \{N_{i,1}, \ldots, N_{i,Y}\}$ where Y is the number of agents Ag_i is able to communicate with; (iii) One Local Policy $LP_i = \{\pi_l\}$ for every agent Ag_i in the system; (iv) A set of Remote Policies $RP_i = \{\pi_{i,j,k}, \ldots, \pi_{i,Y,K}\}$ for each agent in the system, where $\pi_{i,j,k}$ is the Remote Policy corresponding to Local Policy π_k of agent Ag_j, and K is the number of Local Policies of agent Ag_j. If all agents are implementing SP-DWL, $K = 1$ and $|RP_i| = Y$; (v) A Reward Weighting function $F(\boldsymbol{r})$, where $\boldsymbol{r}$ is the vector of rewards for all objectives.

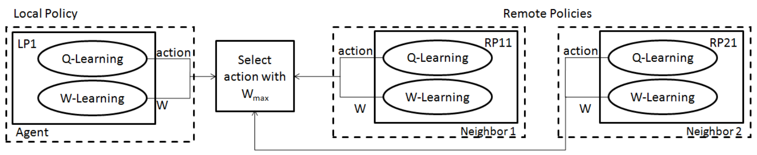


Figure 3: SP-DWL action selection with two neighbors. There is just one Local Policy even if the algorithm is dealing with more than one objective. Dashed rectangles separate which policy refers to which agent (local agent or neighbor). There are a total of 3 policies in this scenario.

The decision process of the winning action remains the same as in DWL; thus, for each time step, the winning action is decided among local and remote policies, as illustrated in Figure 3. When updating the Q and W tables, the reward signal will be the result of $F(\boldsymbol{r})$, defined as:

$$F(\boldsymbol{r}) = \sum_{i=1}^{|\boldsymbol{r}|} w_i \times r_i, \tag{2}$$

where w_i is a weighting parameter for objective i.

Although in this paper $F(\boldsymbol{r})$ is defined by Equation 2, other functions could be used [6]. It is also noteworthy that, since the number of policies is reduced, the size of data in each communication between agents is also reduced, because (if all agents implement SP-DWL) only one reward and one current state must be informed to neighbors for each decision step. The DWL algorithm (Algorithm 1) is executed with the *updateLPValues* procedure of Algorithm 2, i.e., the process for Local Policy update has changed from DWL. SP-DWL has only one Local Policy π_l, and multiple objectives are dealt with a Reward Weighting function, thus, the loop on line 4 of Algorithm 1 is executed only once.

Algorithm 2 updateLPValues procedure in SP-DWL

```
1: procedure updateLPValues($W_{win}$,$a_{win}$,$s_l$)
2:     Observe $\pi_l$'s current state $s'_l$
3:     Get vector $\boldsymbol{r}$ with rewards for all objectives
4:     Calculate $r_{\pi_l} = F(\boldsymbol{r})$ (Equation 2)
5:     update $Q_{\pi_l}$ using $(s_l, a_{win}, r_{\pi_l}, s'_l)$
6:     if $W_{\pi_l} \neq W_{win}$ then
7:         Update $W_{\pi_l}(s_l)$ (Equation 1)
8:     end if
9: end procedure
```

One SP-DWL drawback is that its local policy operates in a state space that is the cross-product of all variable states relevant to all objectives. This is not a problem when the same variables are relevant for multiple objectives or there are not too many variables to describe the current state (as in our domain). However, the state space can become too complex when multiple objectives deal with different state variables. In that situation, the regular DWL is a better option. Another issue is that there is no automatic approach to define the best function $F(\boldsymbol{r})$ for a given problem, which must be defined at the design time.

4 The Smart Grid Problem

The modeling of the Smart Grid load balancing problem for our experiments was based on the experiment described in [11]. Experiments published in [5,12] also used a very similar model.

There are 6 houses connected to a transformer, as illustrated in Figure 1, and each house has an EV that is used every day on a daily journey. Each EV requires a 5-hour charge to fully charge its battery, and an average of 3.5-hour charge is needed to meet its daily journey. The hour for leaving the house is drawn from a normal distribution at a mean of 9h a.m. and a variance of 1h. The journey duration is drawn from a normal distribution with a mean of 12.5h and a variance of 1h.

Each EV has a communication link with the next and the previous EV (the first and last EVs have only one neighbor). In our experiments, the user demand from the houses is not considered, and the transformer is in an overload state if more than two EVs are charging at the same time. At each time step, every EV receives a notification of the transformer load prior to their actions. The actions can be *charge* or *not charge*. The state space for each EV is composed of two variables: (1) **current battery charge -** discretized in slots of 20% of the full charge (i.e., it can assume 5 possible values); and (2) **transformer load -** number of EVs charging on the last time step.

We defined 3 reward functions (3 Policies for DWL), partially based on the reward functions published in [5]; however, we removed the assumption that each EV knows how much battery is needed for a daily journey: ***Reward Function 1***: This reward encodes the need of having enough battery for its daily journey. When the time comes for the EV to perform its daily journey, if there is enough battery, the agent receives a reward of +400. In case the battery does not have enough energy, or no daily journey begin at this time step, the agent receives a reward of 0; ***Reward Function 2***: This reward encodes the user desire for maintaining the EV battery at the highest level possible. The battery is considered to be in a low level if there is remaining less than 20% of the full charge, and in a high level if there is more than 80% of the full charge. When the battery is low and the EV is charging, the agent receives a reward of +400, if the EV is not charging, the reward is 0. If the battery is at a medium level and the EV is charging, the agent receives a reward of +100, while if the EV is not charging, the reward is 0. Finally, if the battery is at a high level and the EV is charging, the received reward is +100, in case it is not charging, the reward is +50; ***Reward Function 3***: The last reward avoids the transformer overload. If the transformer is in an overload status or no car is using the transformer, the agent receives a reward of 0. In case the transformer is in the desired load, the reward is +400.

For DWL, each Reward Function represents one objective to be achieved, thus, each agent has three Local Policies.

5 Experiments and Results

This Section describes our experimental evaluation of SP-DWL through a comparison with other algorithms in the Smart Grid problem.

5.1 Experimental Setup

In order to verify if the proposed algorithm would deal better with the difference in temporal frequency of the rewards, we evaluated 4 different algorithms:

- **Random Policy**: Each EV completely ignores all other EVs and has 50% of probability of being charging at each time step.
- **Q-Learning**: The Q-Learning algorithm was implemented receiving the same reward weighting function as our proposal (Equation 2). All agents learn independently from each other, i.e., this algorithm behaves in the same way as our proposed algorithm if the cooperation parameter is set to $C = 0$.
- **DWL**: Each EV has 3 Local Policies, that will receive rewards as described in Section 4. For this algorithm $C = 1$, i.e., all agents are fully cooperative.
- **SP-DWL**: The agents are implemented as described in Section 3. For this algorithm $C = 1$.

A decision is made for each 15-minute period (i.e., each decision, *charge* or *not charge*, lasts 15 minutes). If the EV has a completely charged battery, it can no longer use the *charge* action. When the time comes for the daily journey of a given EV, it cannot perform any action or update its Q-Table until the car has come back. The ϵ-greed policy was used in the learning phase, and the following parameters were set for all algorithms: $\alpha = 0.3$, $\gamma = 0.9$ and $\epsilon = 0.1$. A greedy policy is used in the execution phase (i.e., the algorithms only choose the best actions from their policies). All algorithms and simulations were implemented in MATLAB [7].

5.2 Experimental evaluation

We evaluate the performance of each algorithm for each reward function. Every algorithm was trained with a learning phase of 5 days, and tested in an execution phase of 10 days, where the cumulative reward for each reward function was stored (only for rewards received in the execution phase). Then, the learning phase duration is incremented by 5 days (i.e., on the second iteration the learning ends on the 10^{th} day), and this procedure is repeated until the learning time totals 100 days. The graphs show an average of 150 repetitions of the experiments. The weight vector for the reward weighting function (calculated by

Equation 2) was defined as $\boldsymbol{w} = [1.0, 0.5, 0.25]$, i.e., the battery reward functions are prioritized over the transformer reward function, since the user comfort is a priority in the Smart grid domain. The same weight vector was used for SP-DWL and Q-Learning algorithms.

Figures 4, 5 and 6 show the results of this experiment for all three reward functions. Since there is plenty of time for all EVs to charge until their daily need, the Random approach presents very good results for the battery policies (Rewards 1 and 2); however, this algorithm is likely to cause the transformer overload in many time steps, resulting in a poor performance for the transformer policy (Reward 3).

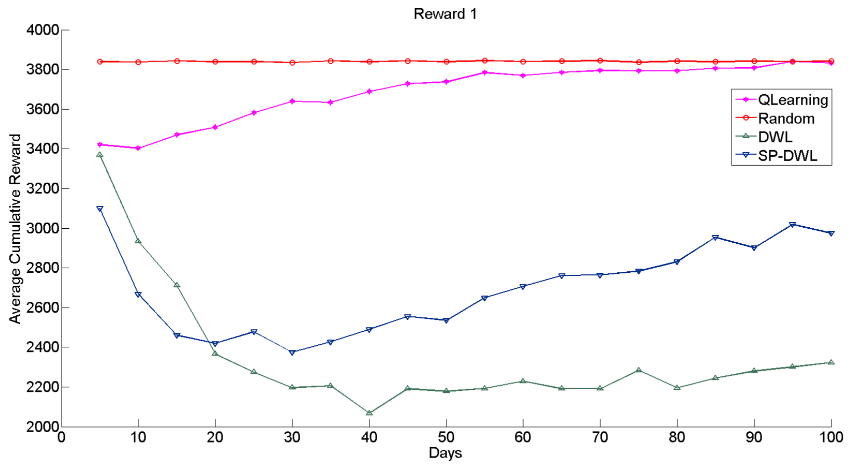


Figure 4: Average cumulative reward for Reward 1 after 150 repetitions of the experiment. The horizontal axis represents days of learning phase, and each point is the average cummulative reward in 10 days of execution phase.

Q-Learning presented one of the best results for the policy learned with Reward 1, and the best result for Reward 2, which was expected since the battery policies were prioritized by the weight vector. However, as the agents are unable to cooperate, Q-Learning had a poor performance for Reward 3, even worse than the Random approach after some time of learning. DWL, as expected, prioritized Reward 3, since this reward function provides a feedback for all decision steps, while Reward 1 gives feedback once each day. DWL achieved good results for Reward 3, yet Figures 4 and 5 show that DWL achieved the worst results for Rewards 1 and 2, and after 30 days of learning this algorithm roughly stabilizes, i.e., this algorithm is not notably improving its performance with more learning time. SP-DWL, in turn, provided a better balancing between the three reward functions. For the battery policies, SP-DWL performance is better than DWL, while maintaining a good performance for the transformer policy. According to a Wilcoxon Signed Rank Test with a confidence level of 95%, the difference

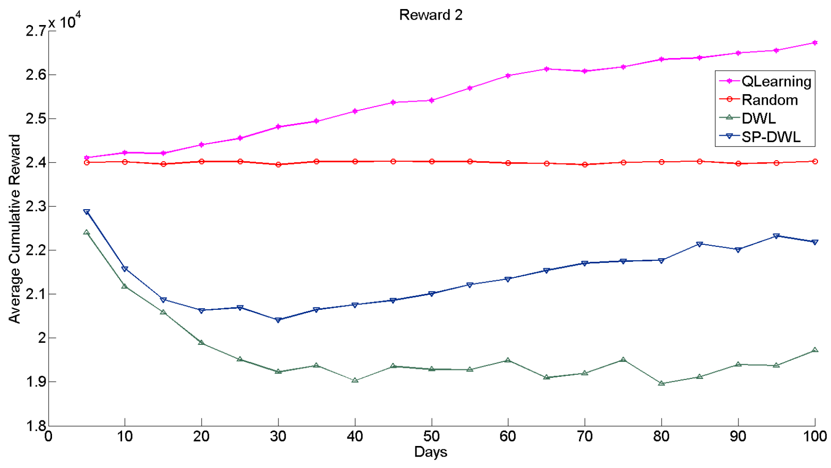


Figure 5: Average cumulative reward for Reward 2 after 150 repetitions of the experiment. The horizontal axis represents days of learning phase, and each point is the average cummulative reward in 10 days of execution phase.

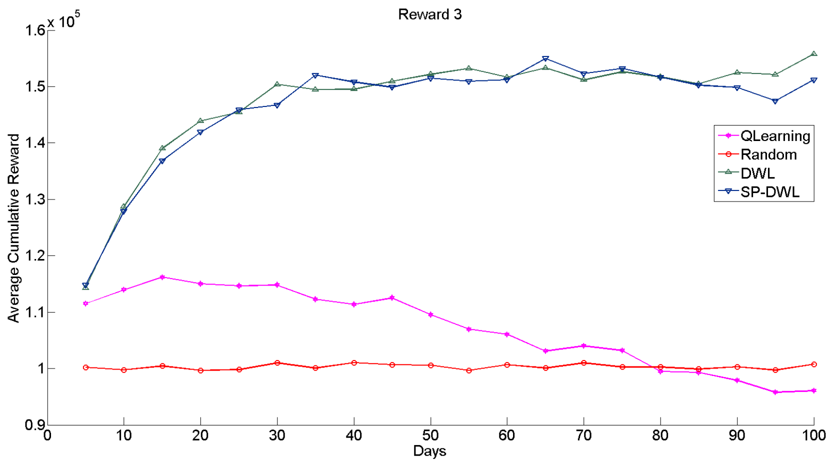


Figure 6: Average cumulative reward for Reward 3 after 150 repetitions of the experiment. The horizontal axis represents days of learning phase, and each point is the average cummulative reward in 10 days of execution phase.

between the average cummulative reward for DWL and SP-DWL is statistically significant in all cases for Reward 1 and Reward 2, except for 20 days of learning (for Reward 1) and for 15 days of learning (for Reward 2). Analyzing these results, SP-DWL is the most suitable algorithm for the Smart grid domain, specially because of the difference in performance between SP-DWL and DWL in Figures 4 and 5.

6 Conclusion

We contributed the Single Policy Distributed W-Learning (SP-DWL) algorithm, a variation of the Distribution W-Learning (DWL) algorithm. In SP-DWL, multiple objectives are combined in one reward weighting function, and each agent has just one Local Policy. Our experiment in the Smart Grid domain showed that SP-DWL can deal with conflicting objectives that provide rewards in different time frequencies better than DWL, while maintaining the agent's cooperation ability. SP-DWL also requires communicating less data per decision step than DWL, since only one current state and one reward must be communicated to neighboring agents. Further works will focus on studying means of automatic detection and definition of more meaningful and informative Reward Weighting Functions. Another open question is if the transfer learning approaches used in DWL [11,12] are adaptable to SP-DWL.

References

1. Bazzan, A.L.C.: Beyond reinforcement learning and local view in multiagent systems. Künstliche Intelligenz 28(3), 179–189 (2014)
2. Buşoniu, L., Babuška, R., De Schutter, B.: A comprehensive survey of multi-agent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics 38(2), 156–172 (Mar 2008)
3. Devlin, S.: Potential-Based Reward Shaping for Knowledge-Based, Multi-Agent Reinforcement Learning. Ph.D. thesis, University of York (2013)
4. Dusparic, I., Cahill, V.: Distributed W-Learning: Multi-policy optimization in self-organizing systems. In: Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems. pp. 20–29 (Sept 2009)
5. Dusparic, I., Harris, C., Marinescu, A., Cahill, V., Clarke, S.: Multi-agent residential demand response based on load forecasting. In: 1st IEEE Conference on Technologies for Sustainability (SusTech). pp. 90–96 (Aug 2013)
6. Marler, R., Arora, J.: The weighted sum method for multi-objective optimization: new insights. Structural and Multidisciplinary Optimization 41(6), 853–862 (2010)
7. MATLAB: version 7.14.0 (R2012a). The MathWorks Inc., Massachusetts (2012)
8. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11(3), 387–434 (2005)
9. Roijers, D., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. Journal of Artificial Int. Research 48, 67–113 (2013)
10. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press, Cambridge, MA, USA, 1st edn. (1998)
11. Taylor, A., Dusparic, I., Galvan-Lopez, E., Clarke, S., Cahill, V.: Transfer learning in multi-agent systems through parallel transfer. In: Proceedings of the Workshop on Theoretically Grounded Transfer Learning (2013)
12. Taylor, A., Dusparic, I., Galvan-Lopez, E., Clarke, S., Cahill, V.: Accelerating learning in multi-objective systems through transfer learning. In: International Joint Conference on Neural Networks (IJCNN). pp. 2298–2305 (July 2014)
13. Vaccaro, A., Loia, V., Formato, G., Wall, P., Terzija, V.: A self-organizing architecture for decentralized smart microgrids synchronization, control, and monitoring. IEEE Transactions on Industrial Informatics 11(1), 289–298 (Feb 2015)
14. Wooldridge, M.J.: An Introduction to MultiAgent Systems (2. ed.). Wiley (2009)

Defining a Continuous Marketplace for the Trading and Distribution of Energy in the Smart Grid

Jesús Cerquides[1], Gauthier Picard[2], and Juan A. Rodríguez-Aguilar[1]

[1] IIIA-CSIC, Campus UAB, 08193 Cerdanyola, Catalonia, Spain
{cerquide,jar}@iiia.csic.es
[2] Laboratoire Hubert Curien UMR CNRS 5516
Institut Henri Fayol, Mines Saint-Etienne, France
picard@emse.fr

Abstract. Decentralized energy production is meant to reduce generation and distribution inefficiencies, leading to major economic and environmental benefits. This new model is meant to be supported by smart grids, electricity networks that can intelligently integrate the actions of all users connected to them —generators, consumers, and prosumers (those that do both)— to efficiently deliver sustainable, economic and secure electricity supplies. A major research challenge is the design of markets for prosumers in smart grids that consider distribution grid constraints. Recently, a discrete market model has been presented that allows prosumers to trade electricity while satisfying the constraints of the grid. However, most of the times energy flow problems possess a continuous nature, and that discrete market model can only provide approximate solutions. In this paper we extend the market model to deal with continuous (piecewise linear) utility functions. We also provide a mapping that shows that the clearing of such a market can be done by means of integer linear programming.

Keywords: smart grid; energy market; prosumers; mixed integer programming

1 Introduction

Our centralized model of production and transmission wastes enormous amounts of energy. According to [6], ”...an astonishing two-thirds of primary energy inputs”. Since power stations are generally far from centers of demand, much of the produced heat is not used, but vented up chimneys or discharged to rivers. Additional losses come about as the electricity travels along the wires of the transmission and distribution systems [6,23]. As argued in [23], favoring the decentralized generation of energy over traditional centralized electricity generation will reduce generation and distribution inefficiencies and will facilitate increased contributions from renewables. This new model is meant to be supported by smart grids.

Following [3], a smart grid is an electricity network that can intelligently integrate the actions of all users connected to it —generators, consumers, and *prosumers* (those that do both)— to efficiently deliver sustainable, economic and secure electricity supplies. In the smart grid the consumer can be either an individual or a household, but also a community or an SME. In its more general form, a smart grid is populated by prosumers capable of both generating and consuming energy. Therefore, smart grids clearly play the central role in the integration of all these prosumers (electricity grid users) by means of the enactment of a system that satisfies a number of societal goals. Out of these goals, there is that of setting market-based prices for electricity taking into account grid system constraints. Thus, a major research challenge in the heart of several roadmaps for the Smart Grid [3,4] is the design of markets for prosumers in smart grids that consider distribution grid constraints. This vision will allow prosumers to directly trade over the smart grid [8]. Following [18], market operations will involve a large number of heterogeneous prosumers, distributed throughout the network (closer to the point of use of electricity), and trading much smaller amounts of energy that are nowadays traded. The distribution of electricity employs one of the three common types of network topologies: radial, ring main, and interconnected [5,7,21]. On the one hand, radial networks are acyclic. On the other hand, as observed in [13], though ring main and interconnected networks contain cycles, they are configured into acyclic networks by means of switches to supply power [7,21].

The smart grid vision has spurred a wealth of research on the design of markets and trading agents for the smart grid. The state-of-the-art has mainly considered to employ different types of auctions for this endeavor. Thus, the market-based trading of energy is typically addressed by the literature by having prosumers participate in a double auction where energy is traded on a day-ahead basis [8,9,10,14,16,20]. Submitted buy and sell orders for energy are matched either by means of either a continuous double auction [10,16,20] or a call market [8,9,14]. Exceptions to this common approach are represented by the tailored multi-unit auctions in [22] and the simultaneous combinatorial reverse auctions employed in [17] to match demand and supply.

In [1], the limitation of the market mechanisms employed in the literature are identified, noticing that up to then, no mechanism takes into account grid system constraints. Thus, the clearing of the market occurs disregarding, for instance, that the transmission of energy is carried out along capacity-constrained distribution networks (which is an actual-world constraint [21]). Therefore, trading and distribution are considered as decoupled activities. Furthermore, the bidding language offered to grid users is pointed out to be not expressive enough to express a prosumer's energy profile since with the exception of [17], which supports combinatorial bids, double auctions limit a grid user to submit a single price-quantity bid to either buy or sell. This does not allow a prosumer to express a full energy profile encompassing a combination of all her buy and sell offers.

As a consequence of this analysis they introduce the Energy Allocation Problem (EAP) as the problem of deciding how much energy each prosumer trades as well as how energy must be distributed throughout the grid so that the overall benefit is maximized while complying with the grid constraints and the prosumers' preferences. On the one hand, they consider that the capacity of the distribution network is limited [21]. On the other hand, since a prosumer can both generate and consume energy, their formulation considers that each prosumer can encode her preferences as a combination of offers to both buy and sell energy. Solving the EAP amounts to clearing a prosumer-oriented market. However, in the EAP, prosumers are limited to bid for discrete amounts of energy. That is, a prosumer can offer to buy either 3 KW for 6c€ or 2 KW for 4c€, but it is not allowed to express that he will buy any amount of energy between 2 KW and 3 KW and that he will be willing to pay 2c€ per KW. In many energy settings, such offers make complete sense and provide a better representation of the prosumer interests when approaching the market. Thus, in this paper we make headway towards the application of these models by extending the EAP so that it allows prosumers to communicate continuous (piecewise linear) utility functions.

More precisely, we make the following contributions:

- We extend the Energy Allocation Problem (EAP) into the continuous energy allocation problem (CEAP). It turns out that the extension is not trivial and requires some mathematical development. We provide some of the results required to deal with piecewise linear functions to represent prosumer preferences.
- We show how to encode the CEAP as a mixed-integer program so that it can be optimally solved for any distribution network topology by means of off-the-shelf commercial solvers such as CPLEX or Gurobi.
- Finally, since the CEAP defines the allocation rule of our market, we also touch upon the design of payment rules that together with our allocation rule can help design a mechanism for our prosumer-oriented market.

The rest of the paper is organized as follows. Section 2 formally defines the allocation rule that we propose to clear prosumer-oriented electricity markets with piecewise linear valuation functions. Thereafter, section 3 shows how to implement the clearing of the market as a mixed-integer program (MIP). Next, section 4 touches upon how to cope with prosumers' strategic behavior, and section 5 concludes and sets paths to future research.

2 The energy allocation problem

The aim of this section is to provide a simple mathematical model for the energy market in a prosumer network, and the allocation rule proposed for that market. We start by providing an example of an energy trading scenario that illustrates the model of prosumers and the model of energy network that we will consider. Thereafter, we provide the allocation rule for that market as the solution to an optimization problem: the continuous energy allocation problem (CEAP).

2.1 Example: energy trading scenario

Figure 1 shows an example of an energy trading scenario involving four prosumers, each one represented by a circle. Each edge connecting two prosumers means that they are physically connected. Moreover, each link is labeled with its capacity, namely with the amount of energy it can transport. For instance, prosumer 1 is connected to prosumer 2, and their link can transport up to 2 energy units. Each prosumer can offer to either buy, sell or transmit energy. The offer of each prosumer is represented as a table next to each prosumer in Figure 1, where each entry in the table represents contains the range of

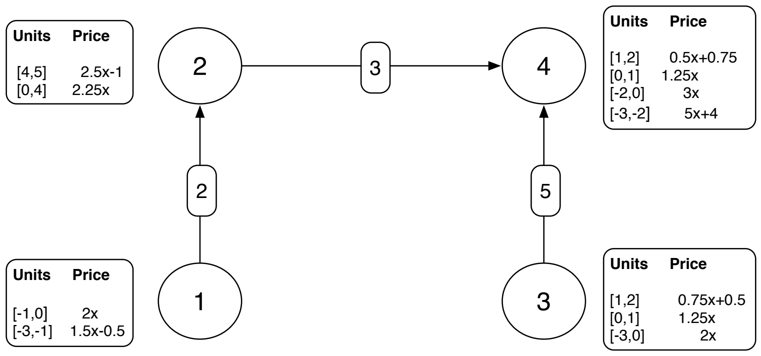


Fig. 1. Energy trading scenario.

energy units to which it applies and the linear function used to obtain the price provided that the prosumer is required to provide a number of energy units in that range. As a convention, a selling offer is expressed by means of a negative number of units, whereas a buying offer is encoded with a positive number of units. For instance, prosumer 4's first entry communicates that, if as a result of clearing the market, he is provided an amount of energy e between 1 KW and 2 KW, he will pay $(0.5 \cdot e + 0.75)$c€. That is if he is provided 1.5KW, he will pay 1.5c€. On the other hand, its last entry states that if he is requested to provide an amount of energy e between 2 and 3 KW, he will be paid $(5 \cdot e - 4)$ c€ (note that the sign is reversed from the expression in the table because we are encoding sell offers with negative numbers). Finally note that, by applying its third valuation, he shows his willingness to transmit energy for free (he will be happy to receive 0KW at price 0c€). In Figure 1, we observe that prosumer 1 only sells energy, and prosumer 2 only buys energy, while prosumers 3 and 4 can either buy or sell.

2.2 Problem definition

Now the problem faced by the prosumers in Figure 1 is to decide how much energy to trade and with whom so that the overall benefit (social welfare) is maximized while the energy network's capacity constraints are fulfilled. This means that: (i) each prosumer must select how much to trade; and (ii) each pair of prosumers connected by a link must agree on the amount of energy to be transferred by their link together with the direction of the transfer (with whom). In what follows we cast this problem as an optimization problem, and we put off the solution to this problem to sections 3.

Following example 1, we consider that the energy network connecting a set of prosumers P can be modeled as an undirected graph (P, E), where the vertexes stand for the prosumers and each edge in E connects a pair of prosumers. An edge $\{i, j\} \in E$ means that prosumer i and j are physically connected to trade energy. When $\{i, j\} \in E$, $i < j$ we say that i is an in-neighbor of j and that j is an out-neighbor of i. The set of in-neighbors (resp. out-neighbors) of j is $in(j)$ (resp. $out(j)$).

Each prosumer j expresses her offers to buy and sell energy by means of an *general valuation function* $o_j : \mathbb{R} \rightarrow \mathbb{R} \cup \{-\infty\}$. For instance, $o_j(3) = 2$ indicates that prosumer j is willing to buy 3 energy units at 2c€, while $o_j(-4) = -2$ indicates that she is willing to sell 4 energy units if paid 2c€. Notice that offer functions capture prosumers' constraints. To communicate her offer function, each prosumer sends a table like the ones in Figure 1 making explicit her feasible energy states and their values. Given a number of units x, if x does not belong to the interval of any of the entries in the table, it means that such energy state is unfeasible for the prosumer and thus its value $o_j(x)$ is $-\infty$. If x appears in more that one interval, then its $o_j(x)$ is the maximum among the values assigned for each of the entries in the table in which it is contained.

In the following we define formally the mathematical foundations that underlie piecewise linear valuations.

Definition 1. *A* general valuation *is any function $\alpha : \mathbb{R} \rightarrow \mathbb{R} \cup \{-\infty\}$. We use F_α to note the subset of $\mathbb{R}$ in which α takes finite values, that is $F_\alpha = \alpha^{-1}(\mathbb{R})$. We define the zero valuation $\mathbf{0}$ as the function that maps every real number to $-\infty$. That is, for all $x \in mathbbR$ we have that $\mathbf{0}(x) = -\infty$. We define the unit valuation $\mathbf{1}$ as the one that maps 0 to 0 and any other element to $-\infty$. That is,* $\mathbf{1}(x) = \begin{cases} 0 & \textit{if } x = 0 \\ -\infty & \textit{otherwise} \end{cases}$

Let $W = \{\omega_1, \ldots, \omega_n\}$ be a finite set of general valuations. We define F_W as the set of values where at least one of the valuations in W takes a finite value. That is, $F_W = \bigcup_{i=1}^n F_{\omega_i}$.

Furthermore, we can define the maximum valuation $\beta = \max W$ as

$$\beta(x) = (\max W)(x) = \begin{cases} \max_{1 \le i \le n} \{\omega_i(x) | x \in F_{\omega_i}\} & \textit{if } x \in F_W \\ -\infty & \textit{otherwise.} \end{cases} \tag{1}$$

Definition 2 (Point Valuation). *A general valuation α is a* point valuation *if and only if F_α contains a single element. We can always represent a point valuation by an ordered pair $(p, q) \in \mathbb{R}^2$, such that*

$$\alpha(x) = \begin{cases} q & \textit{if } x = p \\ -\infty & \textit{otherwise.} \end{cases} \tag{2}$$

Note that the unit valuation is a point valuation represented by the ordered pair $(0, 0)$.

Definition 3 (Linear Interval Valuation). *A real interval is a subset of real numbers $[l, u] = \{x \in \mathbb{R} \mid l \le x \le u\}$. A general valuation α is a* linear interval valuation *if and only if there is a real interval $I_\alpha = [l_\alpha, u_\alpha]$, and two real numbers a_α, b_α, such that for each $x \in \mathbb{R}$*

$$\alpha(x) = \begin{cases} a_\alpha \cdot x + b_\alpha & \textit{if } x \in I_\alpha \\ -\infty & \textit{otherwise} \end{cases} \tag{3}$$

We say that the ordered tuple $(l_\alpha, u_\alpha, a_\alpha, b_\alpha) \in \mathbb{R}^4$ is a representation of α.

Lemma 1. *Any point valuation is a linear interval valuation*

Proof. Let (p, q) be the representation of a point valuation α. Then, $(p, p, 0, q)$ is a representation of α as interval lineal valuation.

Definition 4 (Discrete Valuation). *A general valuation α is a* discrete valuation *if and only there exists a finite set of point valuations $W = \{\omega_1, \dots, \omega_n\}$, such that $\alpha = \max W$. That is, for each $x \in \mathbb{R}$, we have that $\alpha(x) = (\max W)(x)$.*

Definition 5. *A general valuation α is* piecewise linear *if and only there exists a finite set of linear valuations $W = \{\omega_1, \dots, \omega_n\}$, such that $\alpha = \max W$.*

In that case we say that W is a piecewise linear representation *of α of size n. Note that $F_\alpha = \bigcup_{i=1}^{n} I_{\omega_i}$.*

Lemma 2. *Any discrete valuation is piecewise linear.*

Proof. Directly from the definitions of discrete and piecewise linear valuation and Lemma 1.

Note that this means that piecewise linear valuations are a more general framework than that used in [1]. Thus, any algorithm or problem definition that assumes piecewise linear valuations will in particular be capable of working with discrete valuations. Next, we provi

Lemma 3. *Each piecewise linear valuation admits a representation $W = \{\omega_1, \dots, \omega_n\}$ in which*

1. *For each two linear interval valuations ω_i and ω_j, we have that $|F_{\omega_i} \cap F_{\omega_j}| \le 1$. That is, the finite domains of ω_i and ω_j are either disjoint or share a single point.*
2. *There is no point shared by more than 3 linear interval valuations.*
3. *For each $1 \le i < n$ we have that $u_{\omega_i} \le l_{\omega_{i+1}}$.*

We call such a representation a canonical representation*.*

Proof. The proof proceeds constructively. It is relatively simple to build an algorithm that builds the canonical representation of the maximum of two valuations given their canonical representations. On the other hand, for any interval lineal valuation, its canonical representation is direct. Thus given a representation which is not canonical, the canonical representation can always be built by taking the canonical representations of the interval lineal valuations in the representation and then successively taking maximums between them until we have assessed the maximum of all the interval linear valuations in the representation.

Our fundamental assumption in this work is that prosumers' offers are piecewise linear valuations. Hence, in the remaining of the paper when we refer to a *valuation* we will always mean a piecewise linear valuation.

Besides prosumers' offers, we also consider that the energy network is physically constrained by the capacity of the connections between prosumers. We will note as c_{ij} the capacity limit of edge $\{i, j\}$, namely the maximum number of energy units that the link between prosumers i and j can transmit. An allocation specifies the number of units that each prosumer trades with each neighboring prosumer. We will encode an allocation by means of a set of variables $Y = \{y_{ij} \mid i \in P, j \in out(i)\}$, where $\mathbf{y}_{ij}$ stands for the number of units that prosumer i sells to prosumer j and is bounded by the capacity limit c_{ij}. That is, the domain of variable y_{ij} is $D_{ij} = [-c_{ij} \mathinner{..} c_{ij}]$. Thus, if $\mathbf{y}_{ij}$ takes on a value k greater than 0, it means that prosumer i sells k energy units to prosumer j. Otherwise, if $\mathbf{y}_{ij}$ takes on a negative value $-k$, we say that prosumer i buys k energy units from prosumer j. From this follows that $\mathbf{y}_{ij}$ represents a trade from prosumer i's perspective.

Now we want to assess the value of a given allocation. Before that, we will define the local value of a given allocation for a single prosumer. We need to assess the amount of energy that a prosumer acquires and sells according to an allocation $\mathbf{Y}$. Prosumer j will only consider its local view of the allocation, represented by $\mathbf{Y}_j = \mathbf{y}_{\cdot j} \cup \mathbf{y}_{j \cdot}$. We can assess the *net energy balance* for prosumer j as

$$net(\mathbf{Y}_j) = \sum_{i \in in(j)} \mathbf{y}_{ij} - \sum_{k \in out(j)} \mathbf{y}_{jk}, \tag{4}$$

where each $\mathbf{y}_{ij}$ and $\mathbf{y}_{jk}$ are added with different signs because j takes the role of buyer in $\mathbf{y}_{ij}$ and that of seller in $\mathbf{y}_{jk}$. And therefore, the local value v_j of an allocation $\mathbf{Y}$ for prosumer j can be assessed as the value of her net energy balance by means of her offer function

$$v_j(\mathbf{Y}_j) = o_j(net(\mathbf{Y}_j)). \tag{5}$$

Therefore, the value of an allocation $\mathbf{Y}$ can be obtained by adding up the local value of the allocations for each prosumer.

$$Value(\mathbf{Y}) = \sum_{i \in P} v_j(\mathbf{Y}_j). \tag{6}$$

Now, we are ready to define the energy trading allocation as that of finding the allocation of maximum value that satisfies the capacity of the energy network.

Problem 1. Given a set of prosumers P, a canonical representation of their offers $\{o_j | j \in P\}$, and an undirected graph E where each edge is labeled with its capacity c_{ij}, the continuous energy allocation problem (CEAP) amounts to finding an allocation $\mathbf{Y}$ that maximizes $Value(\mathbf{Y})$. Whenever the graph E is acyclic we say that the CEAP is acyclic.

At this point we can consider again the example in Figure 1. When solving the CEAP defined by Problem 1, we obtain the variable assignment shown in Figure 2. The solution indicates that prosumer 1 transfers 2 energy units to prosumer 2 ($y_{12} = 2$), prosumer 2 also receives 3 energy units from prosumer 4 ($y_{24} = -3$), and prosumer 3 transfers 3 energy units to prosumer 4. Next to each offer table, we show the amount of energy $\mathbf{x_i}$ that each prosumer is provided (if $\mathbf{x_i}$ is positive) or requested (when $\mathbf{x_i}$ is negative). This corresponds to the net energy balance (Equation 4). The value of the offer of each prosumer in its energy balance state is added to assess the net value of the allocation (see Equation 5). Thus, the allocation that maximizes Equation 6 has a value of 2.

Notice that prosumer 2 obtains 5 energy units by *aggregating* the energy units received from prosumers 1 and 4. However, prosumer 4 does not sell anything to prosumer 2. The role of prosumer 4 is to *relay* to prosumer 2 the energy transferred from

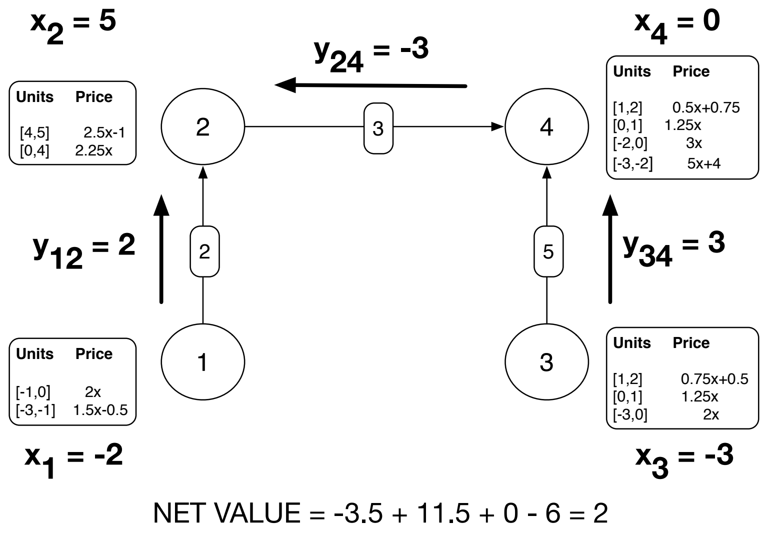


Fig. 2. Solution to the CEAP represented by the energy trading scenario.

prosumer 3, which is the one that does sell energy. In general, our model supports that each prosumer either: (i) aggregates energy received from its neighbors when buying energy; (ii) splits and distributes energy to its neighbors when selling energy; or (iii) relays energy so that other prosumers can satisfy their demand.

3 Solving the CEAP through MIP

Solving optimization problems by mapping them to linear programs has become a standard practice whenever such a mapping can be found. Through the advance of software capabilities (including CPLEX and Gurobi), this practice turns out to be difficult to beat even for problems, such as combinatorial auctions, that have attracted a stream of research in specific algorithms [11]. Along this line, in this section we show how the CEAP can be encoded as a linear program (LP).

Before translating the CEAP as an LP, we consider that the offer of prosumer j is expressed as a piecewise linear valuation o_j. According to lemma 3, each offer o_j admits a canonical representation that hereafter we denote as $W_j = \{o_j^1, \ldots, o_j^{n_j}\}$, where $o_j^1, \ldots, o_j^{n_j}$ are linear interval valuations. Thus, each linear interval valuation $o_j^k \in W_j$ is defined as follows:

$$o_j^k(x) = \begin{cases} a_{o_j^k} \cdot x + b_{o_j^k} & \text{if } x \in I_{o_j^k} \\ -\infty & \text{otherwise} \end{cases} \tag{7}$$

where $I_{o_j^k} = [l_{o_j^k}, u_{o_j^k}]$ is a real interval, $a_{o_j^k}$ and $b_{o_j^k}$ are two real numbers, and $x \in \mathbb{R}$.

To encode our optimization problem, we will consider two types of decision variables: network decision variables and prosumer decision variables. On the one hand, as to the network, as described in section 2, for each edge (i, j) in the trading energy network an integer variable y_{ij} will take on as a value the number of units that prosumer i sells to prosumer j (when $\mathbf{y}_{ij} > 0$), or that she buys from prosumer j (when $\mathbf{y}_{ij} < 0$). Notice that $\mathbf{y}_{ij}$ may also be zero if there is no trading between i and j. In general, the value of y_{ij} is within the domain D_{ij}.

On the prosumer side, since the prosumer value $v_j(\mathbf{Y}_j)$ of equation 5 cannot be encoded as a linear function in terms of these variables, for each prosumer j we introduce a set of auxiliary binary variables $\{z_j^k \mid j \in P,\ 1 \leq k \leq |W_j|\}$, where variable z_j^k indicates whether the k-th linear interval valuation in the offer is taken or not. Since the linear interval valuations within the offer of prosumer j are mutually exclusive, these variables are linked by a constraint that enforces that one and only one of them is active, namely $\sum_{k=1}^{|W_j|} z_j^k = 1$.

Besides choosing some linear interval valuation out of an offer, we must also decide the number of units that the prosumer is to trade. Thus, for each prosumer j we introduce a set of auxiliary real variables $\{x_j^k \mid j \in P,\ 1 \leq k \leq |W_j|\}$, where variable x_j^k indicates the number of units the prosumer decides to trade. Therefore, we can readily encode the value obtained from selecting x_j^k energy units to trade from the linear interval valuation o_j^k as $a_{o_j^k} \cdot x_j^k + b_{o_j^k} \cdot z_j^k$.

At this point, we can establish how to enable each z_j^k variable by means of the following constraint:

$$z_j^k = 1 \text{ if and only if } x_j^k \in I_{o_j^k} \tag{8}$$

This constraint ensures consistency between each prosumer's decisions. If variable x_j^k is set to a value within $I_{o_j^k}$, then variable z_j^k must be enabled to reflect that the k-th linear interval valuation of prosumer j is selected. Thus, each variable z_j^k acts as an indicator variable. Notice that equation 8 can be readily linearised by means of the following inequations: $z_j^k \cdot l_{o_j^k} \leq x_j^k \leq z_j^k \cdot u_{o_j^k}$.

Now we are ready to put together the network and prosumer decision variables. The net energy balance $net(\mathbf{Y}_j)$ from equation 4 provides a connection between the flows of energy in and out a prosumer and the offer selected. We can express equation 4 for prosumer j by means of the constraint

$$\sum_{i<j} y_{ij} - \sum_{q>j} y_{jq} = \sum_{k=1}^{|W_j|} x_j^k.$$

Finally, the prosumer value can be easily written as a linear expression in terms of these variables: $\sum_{l=1}^{|W_j|} v_j^k$, where $v_j^k = a_{o_j^k} \cdot x_j^k + b_{o_j^k} \cdot z_j^k$ is the value contributed by the k-th linear interval valuation.

Now we are ready to define the LP that solves the energy allocation problem introduced in the previous section.

$$\begin{array}{lrl}
\textbf{maximize} & \sum_{j=1}^{|P|}\sum_{k=1}^{|W_j|} a_{o_j^k}\cdot x_j^k + b_{o_j^k}\cdot z_j^k & \\
\textbf{subject to} & z_j^k\cdot l_{o_j^k} \le x_j^k \le z_j^k\cdot u_{o_j^k} & \forall j\in P, 1\le k\le |W_j| \\
 & \sum_{l=k}^{|W_j|} z_j^k = 1 & \forall j\in P, 1\le k\le |W_j| \\
 & \sum_{i<j} y_{ij} - \sum_{q>j} y_{jq} = \sum_{k=1}^{|W_j|} x_j^k & \forall j\in P, 1\le k\le |W_j| \\
 & & \\
 & y_{ij}\in D_{ij} & \forall (i,j)\in E \\
 & z_j^k\in\{0,1\} & \forall j\in P, 1\le k\le |W_j| \\
 & x_j^k\in\mathbb{R} & \forall j\in P, 1\le k\le |W_j|
\end{array}$$

Let us consider again the example in Figure 1, and its solution in Figure 2. The optimal allocation **Y** presented in the previous section is obtained by the MIP above by setting the network decision variables to the following values: $y_{12} = 2$, $y_{24} = -3$ and $y_{34} = 3$; and the prosumer decision variables to the following ones: $x_1^1 = -2$, $x_2^2 = 5$, $x_3^1 = -3$ and $z_1^1 = 1$, $z_2^2 = 1$, $z_3^1 = 1$, $z_4^3 = 1$ (otherwise $x_j^k = 0$ and $z_j^k = 0$). This leads to the following evaluation of the allocation (only those j, k sumands with $z_j^k = 1$ are shown, since all others are zero):

$$[1.5\cdot(-2) - 0.5] + [2.5\cdot 5 - 1] + [1.25\cdot 0] + [2\cdot(-3)] = -3.5 + 11.5 + 0 - 6 = 2$$

4 Mechanism Design

Up to now, we have concentrated on how to formalize and provide a solution to the CEAP through ILP, disregarding the strategic behavior of prosumers. Here we skim through some game-theoretic considerations.

Mechanisms are composed of both a choice rule and a payment rule [19]. From a mechanism design point of view, the CEAP can be understood as the choice rule that selects the energy trades in our network based on the valuations provided by the prosumers. The previous section shows that this choice rule can be assessed by means of ILP However, we have not proposed any payment rule that establishes how much should each agent pay/receive afterwards.

In their classical work from 1983, Myerson and Satterthwaite [15] proved the impossibility of having an efficient, individual-rational, incentive-compatible, and budget-balanced mechanism in a simple exchange environment in which a buyer and a seller trade a single unit of a given good. This very simple case is isomorph to an energy network with two connected participants where one has available an energy unit that the other one wants to buy. Thus, the impossibility result [15] extends to our setting.

On the other hand, the central result in mechanism design, on the incentive-compatibility of the Vickrey-Clarke-Groves (VCG) mechanism, carries over to our model. Recall that the VCG mechanism allocates goods in the most efficient manner and then determines the price to be paid by each bidder by subtracting from their offer the difference of the overall value of the winning bids and the overall value that would have been attainable without that bidder taking part. That is, this “discount” reflects the contribution to the overall production of value of the bidder in question. The VCG mechanism is strategy-proof: submitting their true valuation is a (weakly) dominant strategy for each bidder. As an inspection of standard proofs of this result reveals [12], this does not depend on the internal structure of the agreements that agents make. Hence, it also applies to our model.

Furthermore, assessing the VCG payment for each prosumer only requires solving a new CEAP problem where that particular prosumer is not present, which can also be done by means of generic ILP software such as CPLEX or Gurobi.

Further studying mechanism design properties of such markets (including alternative payment rules that could lead to asymptotic efficiency along the lines of [2]) remains as future work.

5 Conclusions and future work

In this paper we have investigated how to extend the work in [1] to enable energy trading in prosumer networks for prosumers with piecewise linear valuations, and taking into account grid system constraints. We propose to cast the energy trading problem as an optimization problem, the continuous energy allocation problem (CEAP). We then show that the CEAP can be formulated as an MIP so that it can be optimally solved for any network topology by means of commercial optimization solvers.

A solver for the CEAP by means of the mapping provided in this paper has effectively been implemented and is currently able to solve problems with hundreds of prosumers in the order of a tenth of a second. A detailed evaluation of the efficiency of that solver is ongoing.

In [1], an alternative distributed algorithm (RadPro) is provided for efficiently solving the discrete EAP when the graph is acyclic. Another promising line of future work is the extension of RadPro to provide a decentralized solver for the acyclic CEAP. Provided that this is successfully achieved, the next step will be to consider how to extend such a solver so that it is able to effectively solve problems which contain cycles.

Acknowledgments

Work partially funded by the ITEA2 SEAS (Smart Energy Aware Systems, ref.12004), COR (TIN2012-38876-C02-01) and GEAR (201350E112) projects.

References

1. J. Cerquides, G. Picard, and J. A. Rodriguez-Aguilar. Designing a Marketplace for the Trading and Distribution of Energy in the Smart Grid. In *AAMAS*, pages 1285–1293, 2015.
2. L. Chu and Z. Shen. Truthful double auction mechanisms. *Operations research*, 56(1):102–120, 2008.

3. European Technology Platform. SmartGrids SRA 2035. Strategic Research Agenda. Update of the SmartGrids SRA 2007 for the needs by the year 2035, March 2012.
4. Federation of German Industries (BDI). The Energy Industry on the Way to the Internet Age. BDI publication No.439, 2010.
5. T. Gonen. *Electric power distribution engineering*. CRC press, 2014.
6. Greenpeace. Decentralising power: An energy revolution for the 21st century. `http://bit.ly/1xf1RCk`, 2005.
7. L. L. Grigsby. *Electric Power Generation, Transmission, and Distribution*. CRC press, 2012.
8. D. Ilic, P. G. Da Silva, S. Karnouskos, and M. Griesemer. An energy market for trading electricity in smart grid neighbourhoods. In *6th IEEE International Conference on Digital Ecosystems Technologies (DEST), 2012*, pages 1–6. IEEE, 2012.
9. K. Kok, B. Roossien, P. MacDougall, O. van Pruissen, G. Venekamp, R. Kamphuis, J. Laarakkers, and C. Warmer. Dynamic pricing by scalable energy management systems—field experiences and simulation results using powermatcher. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8. IEEE, 2012.
10. S. Lamparter, S. Becher, and J.-G. Fischer. An agent-based market platform for smart grids. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry track*, pages 1689–1696. International Foundation for Autonomous Agents and Multiagent Systems, 2010.
11. K. Leyton-Brown, E. Nudelman, and Y. Shoham. Empirical hardness models. *Journal of the ACM*, 56:1–52, 2009.
12. A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic theory*. Oxford University Press, 1995.
13. S. J. O. Miller. *Decentralised Coordination of Smart Distribution Networks using Message Passing*. PhD thesis, University of Southhampton, 2014.
14. J. Mockus. On simulation of the nash equilibrium in the stock exchange contest. *Informatica*, 23(1):77–104, 2012.
15. R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 1(29):165–281, 1983.
16. M. A. Olson, S. J. Rassenti, V. L. Smith, M. L. Rigdon, and M. J. Ziegler. Market design and motivated human trading behavior in electricity markets. In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, 1999. HICSS-32.*, pages 1–27. IEEE, 1999.
17. Y. K. Penya and N. R. Jennings. Optimal combinatorial electricity markets. *Web Intelligence and Agent Systems*, 6(2):123–135, 2008.
18. S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Putting the 'smarts' into the smart grid: A grand challenge for artificial intelligence. *Commun. ACM*, 55(4):86–97, Apr. 2012.
19. Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.
20. P. Vytelingum, S. D. Ramchurn, T. D. Voice, A. Rogers, and N. R. Jennings. Trading agents for the smart electricity grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 897–904. International Foundation for Autonomous Agents and Multiagent Systems, 2010.
21. B. M. Weedy, B. J. Cory, N. Jenkins, J. Ekanayake, and G. Strbac. *Electric power systems*. John Wiley & Sons, 2012.
22. T. K. Wijaya, K. Larson, and K. Aberer. Matching demand with supply in the smart grid using agent-based multiunit auction. *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–6, Jan. 2013.
23. P. Wolfe. The implications of an increasingly decentralised energy system. *Energy policy*, 36(12):4509–4513, 2008.

Identifying Central Points in Road Networks Using Betweenness Centrality

Rodrigo A. Batista and Ana L. C. Bazzan

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
`{rabatista,bazzan}@inf.ufrgs.br`
`http://www.inf.ufrgs.br`

Abstract. This paper aims to identify central points in road networks considering traffic demands. This identification of points is made with a variation of betweenness centrality metric. In this variation, the graph corresponding to the network is weighted according to the routes generated from the traffic demand. To test the proposed approach three networks have been created, which are Porto Alegre and Sioux Falls cities and a regular 10x10 grid. Then traffic demands were microscopically simulated and the results were compared with the proposed method.

Keywords: Traffic Assignment, Complex Networks, Centrality

1 Introduction

Metropolitan regions are currently facing major problems of urban mobility, which have resulted, mostly, from choosing private transportation rather than public transportation. For example, in 2014 Brazil had 78.1 million private vehicles, which represents an increase of 229.3% of fleet vehicles when we consider the last 10 years. This means one private vehicle per 2.6 inhabitants.

Therefore dealing with such growth in the fleet requires duly planning the highway system along mechanisms and strategies to reduce the effects of traffic on the population and the environment. Planning transportation systems involves, among other factors, analyzing the distribution of traffic flow on road networks. This is an essential activity for dealing with the maintenance of these networks and it may contribute to diminish the effects of traffic.

There is evidence that the measure of betweenness centrality, as proposed by Freeman [7], is capable of predict traffic flows in road networks. However, this measure ought to be adapted because it does not suitably represents the distribution of demand [11, 9, 8]. The objective of this paper is to analyse how the measure of betweenness centrality may be adapted and used in identifying central points in a road network. In this paper the term central points refers to those points that are most often traversed by road users upon traveling along their routes.

2 Definitions

2.1 Transport Networks

Transport networks can be considered networks that are composed of roadways and junctions between roadways (*e.g.* intersections). Such networks are typically represented as weighted directed graphs $G = \{V, E\}$, composed of a set of vertices V (junctions) and edges E (roadways) and a cost function *C(e)* which associates weight with each edge. In context of public transport networks, the length, the travel time or the capacity are commonly used as the weight of the edges.

Demand for traffic represents the behavior of users in using the network infrastructure. By behavior, it is understood that the decisions made by the users that are relevant to the problem which is being modeled (*e.g.* choice of routes).

The locations of origin and destination of demand are added in districts. Districts can be defined based on information obtained through sociodemographic studies, data of georeferencing and urban statistics, so that these variations may be the least possible within a given district [1]. In general, and also in this work, each district is associated to a location within the network and it is composed of a set of vertices and edges without there being overlapping on other districts.

The demand of a network is commonly represented by a matrix that relates districts of origin and destinations, associating each of these possible combinations, to a figure that corresponds to the intensity in which these trips occur. As it relates origins and destinations, it is called an origin-destination matrix, or OD matrix.

A driver who wishes to travel from district s to district t, represented by *(s,t)*, may encounter more than one series of edges that lead from s to t. Each of these possible paths is called a route. Since of each of these edges has an associated cost, there is particular interest in the path with the lowest cost, which in this article is called the path with the least length, or the shortest path. Therefore, creating the routes consists of associating series of edges to trips that are specified by the OD matrix.

2.2 Betweenness Centrality

Betweenness centrality is based on the idea that a vertex is the most central as more low cost paths pass through it. The shortest paths between all the pairs of vertices in the network are considered in this calculation. The traditional method for calculating betweenness centrality, as well as another centrality measures, were originally developed in the scope of studying social networks, and they have recently been highlighted in the literature [10].

Betweenness centrality for vertex u is defined in accordance with Equation 1, in which σ_{ij} is the number of shortest paths between i and j, and $\sigma_{ij,u}$ is the number of these paths through which u passes. The shortest paths are calculated on the basis of cost of the edges, we remark that the centrality measure is sensitive to the function of cost that has been chosen.

$$B_u = \sum_{i \in V} \sum_{j \in V \setminus \{i\}} \frac{\sigma_{ij,u}}{\sigma_{ij}}. \tag{1}$$

3 Methods

This section presents a method for calculating betweenness centrality by taking into consideration the demand that is applied to the network. The method consists of constructing a graph that represents the network and weighing it by using the demand. In order that upon calculating betweenness centrality, the occupation of the roadways is taking into consideration, resulting from the demand that is represented by the OD matrix.

3.1 Attributing Weights to the Edges

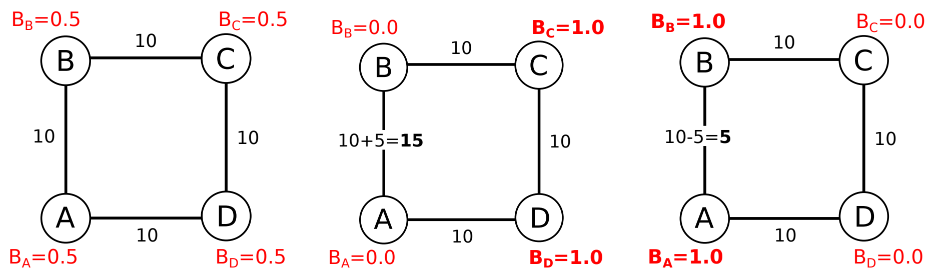
Before calculate betweenness centrality by considering the demand, is necessary to determine the routes that correspond to the demand. As an OD matrix determines only origins and destinations, calculating the routes is necessary for obtaining a table that adds up the amount of routes that pass through each edge. The routes were obtained by calculating the path of the lowest cost between the origin and the destination of each trip that is present in the OD matrix. That is why sequences of edges that form the paths with the lowest cost were found for each OD pair, by using the Dijkstra algorithm [6].

Once the routes that correspond to the demand have been calculated, it is possible to construct a table of occurrence of the amount of routes which pass through each edge. This procedure is a basic stage for defining the weights of the edges of the network graph, since the amount of routes will as input for the functions used to calculate the cost of the edges. Attributing weight to the edges can be carried out in distinct manners. Several studies use the length of the roadway as the cost of the edges [9, 4, 11, 5]. In this paper we have used the occupation rate of the roadways as the weight of the edges, as it is understood that it reflects the use of the network related to the demand. Furthermore, the unitary function, in which the lowest cost routes refer to the number of hops necessary to go from the origin to the destination, was also considered.

In this study, it was decided to use decreasing cost functions to perform the attribution of costs to the edges. Figure 1 illustrates the situation that brought about this decision. A network through which 10 routes pass along each of the edges is shown in Figure 1a; the values of betweenness centrality for each of the vertices are shown in the same figure. On may notice that the betweenness values are equally distributed since this is a regular network.

Supposing that the number of trips between vertices A and B is increased by 5 trips, the natural logic is to increase the number of trips on the edge to 15. Figure 1b illustrates whats happens to the values of betweenness centrality when the cost of edge AB is increased. In this case, as the betweenness centrality algorithm considers the paths of lowest cost, paths that previously passed through AB have ceased to do so. Thus, vertices C and D have come to receive the greatest betweenness values, while vertices A and B were those that received a real increase in demand. This would require the reader to use and inverted interpretation of the measure, so that the vertices with the lowest betweenness values are central in relation to the demand.

In order to solve this problem and make the greatest values of betweenness centrality be attributed to the vertices with the greatest volume of demand, it was decided to use decreasing cost functions. In this case, an increase of demand between vertices A and B causes a decrease in the weight of the edge, as this was attributed by means of a decreasing function. Figure 1c shows that in this case the vertices with the highest values of betweenness coincide with the vertices that have received the highest demand.



(a) Betweenness centrality distribution over network topology.

(b) Betweenness centrality distribution after increasing the cost of AB edge.

(c) Betweenness centrality distribution after decreasing the cost of AB edge.

Fig. 1: Example of the influence of cost of the edges in distributing betweenness centrality.

Thus, the experiments were guided by taking the following cost functions into consideration:

F_1: **Decreasing Exponential** It is possible to use an exponential function for modeling the cost of an edge according to its occupation. Likewise, assuming that the cost of an edge also decreases exponentially related to its rate of occupation, the cost attributed to the edges is defined as function C, as defined in range $(0;1]$. This weight is calculated in accordance with the decreasing exponential function that is shown in Equation 2, a particular case of the family of equations $y = a(1-b)^x$, and it only considers the amount of trips n that passes through a given edge.

$$C(n) = (1 - 0.001)^n. \tag{2}$$

F_2**: Rational Function** The rational cost function shown in Equation 3 was also considered in the experiments. This function was chosen to explore the behavior of betweenness centrality distribution when cost decreases faster than the exponential function previously explained.

$$C(n) = 1/n. \tag{3}$$

F_3**: Decreasing Linear Function** The linearly decreasing function exhibited in Equation 4, in which $k > n$, was also considered in the experiments. The main objective of using a linear cost function is to study the behavior of the proposed method when edge costs are diminishing more smoothly than the decreasing exponential function.

$$C(n) = k - n. \tag{4}$$

F_4**: Number of Hops** The unitary cost function was added to the experiments and it considers the number of hops that were performed. This function represents how many edges there are in the path with the lowest cost that is calculated for an OD pair.

F_5**: Length of an Edge** Considering that several studies have used the length of a roadway as weight of the edges, this was used in the cost function F_5, as a way of comparing this study to previous studies.

3.2 Calculation of Betweenness Centrality Considering Demand

After the graph that represents roadways was constructed and its edges were properly weighted by using the routes that were generated from the OD matrix and cost function, it is possible to calculate the betweenness centrality. As the interest here lies in identifying the central points, the results of the betweenness centrality algorithm will show high values for vertices that have greater demand.

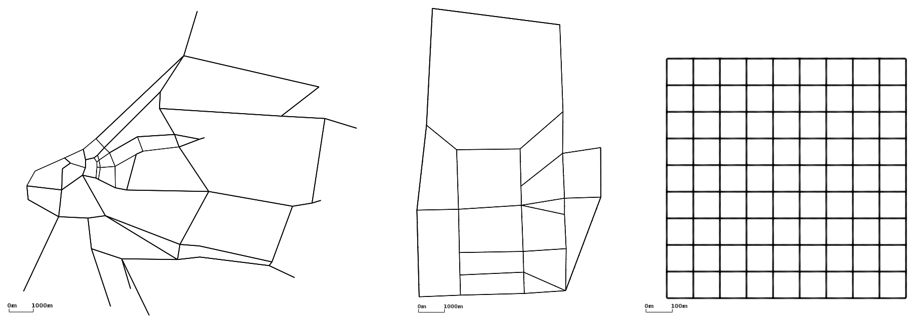
Algorithm 1 lists the steps involved in calculating betweenness centrality considering the demand.

Algorithm 1 Betweenness centrality considering demand

```
Input: Network R, OD Matrix M, Cost Function C
Output: Betweenness centrality from all vertices
1: procedure DemandBetweennessCentrality($R, M, C$)
2:     Construct graph G with the same topology as R
3:     Calculate the routes from M over G
4:     Calculate a table T from the occurrence of routes through each edge
5:     Attribute costs to the edges of G by using T and C
6:     Calculate betweenness centrality over G
7:     return betweenness centrality from the vertices
```

4 Experiments and Results

The proposed approach in this study was tested by means of experiments on three networks. Two of these are abstractions of real networks in the cities of Porto Alegre and Sioux Falls, whereas a third network consists of regular 10x10 grid. All three networks are shown in Figure 2. Furthermore, with the aim of analyzing the behavior of betweenness centrality at different occupation levels, demands with volumes of 10%, 25%, 50% and 75% of the total capacity of each network are used.



(a) *POA Arterials* (b) Sioux Falls (c) 10x10 Grid

Fig. 2: Networks used in the experiments.

4.1 Traffic Demand

For the POA Arterials network a pattern of demand was specified with the aim of reproducing the flow patterns of drivers that are observed in the city of Porto Alegre at the beginning of the day, in which they leave the outskirts of the city and go downtown. In this demand, which is called Non-Uniform Outskirts→Downtown Demand (NUODD), seven distinct points on the outskirts of the city and one point in the central region of the city were used as origin and destination respectively. Considering that the capacity of the POA Arterials network is 127,320 vehicles, demands of volume of 10% (12,372 trips), 25% (31,830 trips) and 50%[1] (63,660 trips) were defined.

For the Sioux Falls network, the same model of demand used in the paper by Chakirov and Fourie [3] was used. In their study the authors based their work on census data to create a de-aggregate demand and a microscopic model of the

[1] The total volume of demand if equal to 50% of the maximum capacity of the network, corresponding to a situation that is much greater than reality. Demands with volumes over 50% higher were not simulated.

Sioux Falls network, based on the network that was originally used in the study by LeBlanc *et al* [12]. In this study only the volume of demand corresponding to the morning rush hour was used. Thus, the demand used in this network has a volume of 44,652 trips and it was generated by an iterative model in order to achieve the stochastic user equilibrium. See [3] for details.

For the regular 10x10 grid two regions were defined, which are the edge and the center, on which three types of demand were defined. The first of these, uniform demand (UD) shows uniform distribution of the origins and destinations of the trips that were generated, and the aim of which if to create random trips within 10x10 grid. The second one Non-Uniform Edge→Center Demand (NUECD), is composed of trips that go from the perimeter toward the center of the grid, and which have the aim of creating congestion in the central region. The third type of demand which if called Non-Uniform Center→Edge Demand (NUCED), is composed of trips that have a pattern that is opposite the previous demand, in other words, trips that leave the center and go toward the edge of the network. Considering that the grid has a capacity of 4,890 vehicles, demands of volume equal to 10% (489 trips), 25% (1,223 trips), 50% (2,445 trips) e 75% (3,668 trips), are used.

4.2 Comparing the Proposed Method to a Microscopic Simulation

Since access to the real measurements that were carried out on roadways of the cities of Porto Alegre and Sioux Falls were not available, it was decided to test the proposed technique by means of comparing it to a simulation. In this case, a microscopic simulation that was performed in the SUMO [2] simulator was used, and it applies the routes that were calculated from an OD matrix to a given network. Figure 3 show the steps involved in the microscopic simulation process and the steps of the proposed method. Both methods receive as input the OD matrix and the road network files and calculate the betweenness centrality of the vertices at the end. For the microscopic simulation, additional steps for trips and routes generation are required to produce the SUMO related files.

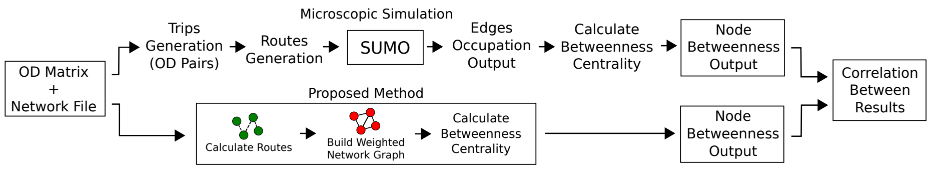


Fig. 3: Steps involved in the betweenness centrality calculation using the proposed method and the microscopic simulation.

By using SUMO it was possible to obtain information about occupation of the edges at the peak of the occupation of the network. Figure 4 shows the mean occupation curve of the Sioux Falls network along time, highlighting time-step of the peak of mean occupation. The rates of occupation obtained were used to weight a graph that represents the network, on which betweenness centrality that serves as a basis for comparing it to the model proposed was calculated. Table 1 shows the vertex-by-vertex details of betweenness centrality that were calculated by the proposed method and the betweenness centrality values calculated at the peak of occupation of the network. The five most significant values of each case study were highlighted to make visualization easier.

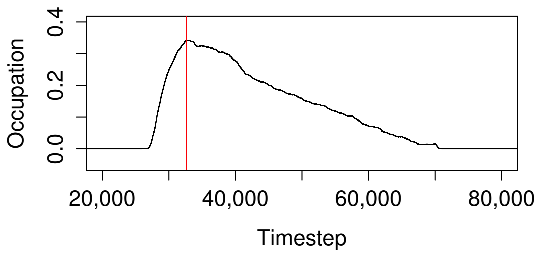


Fig. 4: Mean occupation of the Sioux Falls network related to time, with the peak of occupation occurring at the time-step 32,671.

4.3 Results

With the aim of comparing the results of betweenness centrality obtained by the proposed method to the results obtained through simulation, Pearson's correlation coefficient was used. Thus, the correlation between the results generated by the proposed method with the results obtained by the simulation were calculated for each of the experiments. This correlation was calculated between the results of betweenness centrality over each set set of vertices. Table 2 shows the results that were detailed by the experiments and the cost function.

In the case of the POA Arterials network, decreasing linear function and rational function showed the best results. Considering that hops and edge length functions disregarded demand, it is possible to note that even so, the former showed results that were significantly better than the latter. It is also possible to note that, for lower volumes, the correlation values obtained were greater, which may be attributed to the fact that microscopic simulation considers factors which the static model does not consider.

In the experiment on the Sioux Falls network, the decreasing exponential functions and the rational functions were those that showed the best results. This experiment was the one that showed the lowest rates of correlation. This can be attributed to the fact that the routes of this demand were generated by a different process than the others. In this case, the routes were generated by

Table 1: Betweenness centrality calculated by using different functions for attributing cost and the simulation over the Sioux Falls network, detailed by vertices. The top five betweenness values are shown in red for each case.

Vertex	Cost Function					Simulation
	F_1	F_2	F_3	F_4	F_5	
...	...	...	...	...	...	...
4	19294	17562	17038	19117	17006	12907
5	6554	10559	15610	18557	24134	7572
6	15091	18700	15519	15242	16639	14279
7	3326	3560	3560	3560	2767	10133
8	18790	18808	12784	12961	13774	16894
9	13674	13623	19573	20462	27655	14953
10	42980	37882	36784	30539	33664	31824
11	35269	33075	31314	25068	23418	42001
12	16393	16895	11209	11097	8809	17374
13	10899	8111	5499	5499	4923	4923
14	4657	6185	9400	12623	15148	29303
15	16300	15283	17327	18509	17621	29817
16	25777	21679	12148	12148	13699	29798
...	...	...	...	...	...	...

the model developed by Chakirov and Fourie [3], while in the other case studies, the routes were calculated by considering the shortest paths.

In the case of the 10x10 grid, hops and edge length functions were the ones that showed the best results. Specifically for this case study, the fact that shortest paths with same value exist between a given origin and destination showed a deviation that may have distorted the results. Another point to be noticed is the strong correspondence between the hops and edge length columns, this is due to the regularity of the grid, which makes the edge length function be equal to the hops function.

Therefore, it was observed that the decreasing exponential function and the decreasing linear function showed the best results when the instances of Sioux Falls and POA Arterials were considered. As the occupation peak of the network if being considered, many edges have occupation rates that are near 1, in the hops function also showed significant results, exceeding the others in some cases.

5 Related Work

In Holme's paper [10], the author investigates the relation between traffic flows in communication networks and centrality measures. In this model, particles are moved along the edges of a graph, constrained by the restriction that two particles may not occupy the same vertex at the same time. The particles move along between their randomly defined origins and destinations, and therefore three different updating policies are considered, which are: random walk, in which particles randomly choose a position; detour-at-obstacle, in which a particle randomly

Table 2: Correlation between betweenness centrality calculated by the proposed method and simulation, detailed by experiments and cost functions.

Instance	Cost Function				
	F_1	F_2	F_3	F_4	F_5
Sioux Falls	0.61	0.61	0.59	0.54	0.45
POA Arterials NUODD Vol. 10%	0.96	0.92	0.99	0.98	0.69
POA Arterials NUODD Vol. 25%	0.80	0.84	0.84	0.81	0.58
POA Arterials NUODD Vol. 50%	0.76	0.78	0.77	0.74	0.69
Grade 10x10 NUECD Vol. 10%	0.84	0.79	0.88	0.89	0.89
Grade 10x10 NUECD Vol. 25%	0.90	0.83	0.91	0.86	0.86
Grade 10x10 NUECD Vol. 50%	0.91	0.85	0.93	0.80	0.80
Grade 10x10 NUECD Vol. 75%	0.94	0.80	0.93	0.73	0.73
Grade 10x10 UD Vol. 10%	0.67	0.61	0.67	0.89	0.89
Grade 10x10 UD Vol. 25%	0.63	0.59	0.63	0.91	0.91
Grade 10x10 UD Vol. 50%	0.73	0.73	0.73	0.75	0.75
Grade 10x10 UD Vol. 75%	0.69	0.71	0.67	0.87	0.87
Grade 10x10 NUCED Vol. 10%	0.90	0.73	0.90	0.88	0.88
Grade 10x10 NUCED Vol. 25%	0.84	0.82	0.85	0.87	0.87
Grade 10x10 NUCED Vol. 50%	0.78	0.67	0.78	0.83	0.83
Grade 10x10 NUCED Vol. 75%	0.90	0.77	0.89	0.81	0.81

chooses a position among their neighbors that are nearest the destination; and wait-at-obstacle, in which if no vertices are free near the destination, the particle does not move.

In order to monitor the traffic density regarding betweenness centrality, the author chose the scale-free network model of Barabási-Albert, because it shows a wide distribution of betweenness centrality values. Regarding betweenness centrality, the author noted that the vertices with low or average betweenness rates showed constant occupation rates, and concluded that betweenness centrality itself cannot estimate the capacity of a vertex. At this point, our work differs from Holme's work since we use a microscopic simulation to compare with the betweenness centrality. We also consider nonuniform demands that differ from randomly defined OD pairs used in Holme's work. Beyond that, the subject of study in our work was road networks, while in Holme's work the author focused on communication networks. It influences basically on the network types studied: communication networks could be explained by scale-free models, while road networks could be better explained by random graphs.

In the study of Kazerani and Winter [11], the issue related to the capacity of betweenness centrality for explaining traffic flows was analyzed. In their study, they came to the conclusion that the traditional betweenness centrality measure [7] is unable to explain traffic flows significantly because it does not consider the traffic demand that flows in a network, nor the dynamics.

Still regarding the study of Kazerani and Winter, the authors suggest that an adaptation of the traditional centrality measure is necessary, in which the physical and temporal aspects would be considered, so that a significant correlation with the traffic observed in the network may be attained. In this work, we proposed a variation of the betweenness centrality that, by considering traffic demand, achieved higher correlation with the observed traffic in microscopic simulation.

In the study of Gao *et al* [9], the authors investigate the capacity of betweenness centrality to preview traffic flows by analyzing the correlation between the centrality measure and real traces collected from GPS. For that, data collected from GPS installed on 149 taxis of the city of Qingdao (China) were used.From this analysis, the authors concluded that the betweenness centrality measure does not explains well the traffic flows, and they attribute it two main reasons: first, when calculating betweenness centrality, origins and destinations of trips are vertices of the graph, while in real OD pairs origins and destinations are associated with edges; second, the OD pairs distribution is not uniform, being associated to the distribution of human activity, that is influenced by factor like area occupation.

In a trial to explain the differences between betweenness centrality and traffic flows, the authors established a comparison between the centrality measure and a model developed in three steps: in the first step it is assumed that the demand occurred uniformly over geographic space; in the second, the model was extended to consider the distribution of human activity using data collected from use of cellphones; in the third step, the demand model developed in previous step was extended to support distance decay factor, that models the behavior of people of seeking for resources in nearest places.

When comparing with betweenness centrality values, the model developed in the second step shown greatest correlation. The study suggests that the betweenness centrality presents low correlation with model developed in the third step because it does not consider distance decay factor. However the authors does not suggest modifications in the betweenness centrality, that is known by considers uniform demand and disregards distance decay factor.

In this sense, we also addressed the problem of distance decay factor found in model presented by Gao *et al* by using an OD matrix. Since the OD matrix represents in fact the traffic demand, only the OD pairs related to desired trips will be found in this matrix.

In the study of Galafassi and Bazzan [8] a betweenness centrality variation that considers traffic demand is suggested. Different from the metric proposed by Freeman [7], in their study only the routes that belongs to the specified traffic demand were considered. The authors compare the correlation between the modified measure betweenness centrality with the amount of waiting vehicles on the edges, and show that the proposed method explains traffic flows better than the original measure. The experiments were executed over a 6x6 regular grid and Porto Alegre network, both considering different demand volumes and types (uniform vs nonuniform).

The method proposed here is an extension of the study developed by Galafassi and Bazzan. In this work we kept the betweenness centrality calculation module unchanged, modifying the way of how weights are attributed to the edges to consider the traffic demand. We also compare the proposed method with the occupation of edges during simulation, instead of the waiting vehicles' queue and extended the experiments to consider Sioux Falls network.

6 Conclusions and Future Work

The problem addressed along this work was central points identification in road networks using betweenness centrality. We noticed that some authors tried to explain traffic flows using betweenness centrality and failed because this metric itself assumes a uniform distribution of demand. Thus, our method consisted in combine the betweenness centrality algorithm with the traffic demand so that higher values of betweenness centrality were attributed to the vertices with higher demand.

The proposed method was tested in three networks and a microscopic simulation was performed for each one of them. The occupation of the edges was extracted from the simulations and the results were correlated with the betweenness centrality values calculated by the proposed method. In general, the exponential decay and linear cost functions showed the best results among the studied functions.

The improvement caused by the proposed method was basically caused by two factors. First, the shortest paths calculated by the betweenness centrality algorithm were influenced by the demand that uses a route. The second point is credited to the use of decreasing cost functions, which caused the weight of an edge decrease as a function of the number of routes that pass through it.

Despite the technique proposed in this study being able to help identify central points in transport networks, it is only a step towards a larger goal, which is to improve the road users' travel times. Thus, a possible extension of this study would be to assess whether traffic light operations at these points would improve the average travel time for drivers, and whether the points identified with higher values of betweenness centrality are, in fact, the most critical ones.

Another aspect that could be investigated is to use a weighted correlation coefficient that is calculated considering the capacity of each vertex. A vertex capacity could be estimated by the capacities of its incident edges. Thus, vertices that have large capacity would receive greater weight in the calculation of the correlation.

Another possible extension of this work is to analyze the occupation of links individually during the microscopic simulation, and to approximate a function that models its behavior. This function could be used in the algorithm proposed in this work so that its performance could be compared with other functions considered here.

References

1. Bazzan, A.L., Klügl, F.: Introduction to intelligent systems in traffic and transportation. Synthesis Lectures on Artificial Intelligence and Machine Learning 7(3), 1–137 (2013)

2. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: Sumo-simulation of urban mobility-an overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation. pp. 55–60 (2011)
3. Chakirov, A., Fourie, P.: Enriched sioux falls scenario with dynamic and disaggregate demand. Tech. rep., Working Paper, Future Cities Laboratory, Singapore-ETH Centre (SEC), Singapore (2014)
4. Crucitti, P., Latora, V., Porta, S.: Centrality in networks of urban streets. Chaos: an interdisciplinary journal of nonlinear science 16(1), 015113 (2006)
5. Derrible, S., Kennedy, C.: Applications of graph theory and network science to transit network design. Transport reviews 31(4), 495–519 (2011)
6. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik 1(1), 269–271 (1959)
7. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry pp. 35–41 (1977)
8. Galafassi, C., Bazzan, A.L.C.: Analysis of traffic behavior in regular grid and real world networks. In: The Fifth International Workshop on Emergent Intelligence on Networked Agents (WEIN) (2013), `www.inf.ufrgs.br/maslab/pergamus/pubs/GalafassiBazzan2013-wein.pdf`
9. Gao, S., Wang, Y., Gao, Y., Liu, Y.: Understanding urban traffic-flow characteristics: a rethinking of betweenness centrality. Environment and Planning B: Planning and Design 40(1), 135–153 (2013)
10. Holme, P.: Congestion and centrality in traffic flow on complex networks. Advances in Complex Systems 6(02), 163–176 (2003)
11. Kazerani, A., Winter, S.: Can betweenness centrality explain traffic flow. In: Proceedings of the 12th AGILE International Conference on GIS (2009)
12. LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P.: An efficient approach to solving the road network equilibrium traffic assignment problem. Transportation Research 9(5), 309–318 (1975)

Decentralized allocation of tasks with costs changing over time

James Parker[1], Alessandro Farinelli[2], and Maria Gini[1]

[1] Computer Science and Engineering, University of Minnesota
jparker@cs.umn.edu and gini@cs.umn.edu
[2] Department of Computer Science, University of Verona
alessandro.farinelli@univr.it

Abstract. We propose a decentralized approach for agents to complete tasks with increasing costs over time. Our model accounts for both the natural growth of tasks and the effort of agents at containing such growth. We use this model to reason spatially and temporally to efficiently coordinate agents, i.e., to produce solutions that minimize the growth of tasks. We propose a distributed coordination algorithm (based on max-sum) that is resistant to noise from the environment and shown to outperform state-of-the-art methods from the literature in both a simple simulation and the RoboCup Rescue agent simulation.

1 Introduction

Wide area surveillance, search and rescue, transportation and exploration and mapping all benefit from efficient task allocation. Transportation industries plan distribution routes by solving vehicle routing problems [16] to reduce costs. Our work extends task allocation to cover problems where the costs for completing tasks change over time. Application for this type of problem include minimizing damage from an invasive species, resource distribution for fighting epidemics and containment of forest fires.

In most practical applications, multiple agents need to cooperate to efficiently complete all the tasks. (i.e., fire fighters that cooperate to extinguish large fires). When task costs grow, if too few agents are assigned to a task, it can grow indefinitely. Although we do not consider tasks that have strict deadlines, if the growth rate of a task surpasses the reduction all the agents can provide, then this task can no longer be completed.

A centralized method for allocating tasks that grow over time is given in [10], which here we extend to an efficient decentralized method (binary max-sum). We modify max-sum [3] by incorporating uncertainty into the task growth model, which substantially increases the performance when there is noise or uncertainties from the environment. This provides the first decentralized solution to the problem of task allocation when the cost of tasks grows over time. An approximation of that growth function is assumed to be known upfront, using the model proposed in [10], but new tasks of any size can appear at any time and agents may fail.

In particular, this paper provides the following contributions to the state of the art: (i) a novel formulation of max-sum that incorporates a model for the noise from task cost growth to provide better assignments, (ii) empirical evaluation and comparison with previous state-of-the-art methods in simple simulation and RoboCup Rescue [4], where our formulation of max-sum outperforms previous approaches.

2 Related Work

Multi-agent task allocation with temporal constraints has been researched from a variety of angles. Hard deadlines, also called time windows, are when tasks can be completed only within a specified start and end time. Melvin et al. [8] describe efficient auction based methods if the time windows and rewards for task completion are known. In their work agents bid using local information only, but in our case the rewards are dependent on the actions of other agents due to the cost changing over time. Amador et al. [1] use Fisher market clearing with soft deadlines to assign agents to tasks with rewards that decrease over time. Time in our work has a different effect on tasks by making them harder to complete rather that worth less.

In [13] the authors address tasks that need to be completed before a deadline and provide a decentralized solution using max-sum. Agents reduce the task cost at discrete time intervals similarly to our problem, but they assume task costs are fixed, while we assume task costs change over time. In [12], the authors use an efficient implementation of max-sum for tasks that change over time, but the rewards are purely reactive to a changing environment unlike our approach which can anticipate how the tasks will grow over time. We use the task growth model from the centralized heuristic solution in [10], and propose an efficient decentralized approach.

We model the problem as a Distributed Constraint Optimization Problem (DCOP). DCOP solutions with max-sum can incorporate uncertainty in multiple ways. Stranders et al. [14] show how the multi-armed bandit problem can be used to learn the originally unknown utility, or constraint functions. Unfortunately we cannot learn the utility for our problem, as the utility is dependent on the task size which, in our case, can grow or shrink. Incorporating uncertainty into the DCOP formulation can be done via introducing random variables nodes in the constraint graph not controlled by agents [7] or directly into the utility [2]. Both of these approaches assume the uncertainty is localized in the graph, where in our case uncertainty effects the whole graph. Instead of passing identical uncertainty distributions around inside of the DCOP formulation, we handle the uncertainty outside of max-sum. This keeps the factor graph simple and reduces the number of messages needed to be passed without detracting from the solution quality.

3 Problem Definition

Our problem focuses on the assignment of identical homogeneous agents to tasks which have a cost that changes over time. We make no assumptions on the spatial locations of agents or tasks other than an agent must be on a task's location in order to work on that task. In order for our methods to be effective, there should be more agents than tasks since we assume multiple agents must be assigned to complete a single task. This assumption is not too restrictive. As we show later in Section 5, one can define a task as a cluster of smaller subtasks in order for this property to hold.

We denote the set of identical homogeneous agents by $A = \{a_1, \ldots, a_{|A|}\}$ and the set of tasks by $B = \{b_1, \ldots, b_{|B|}\}$. As mentioned above, $|A| > |B|$. The set of active agent assignments is denoted by $N^t = \{n_1^t, \ldots, n_{|B|}^t\}$, where n_i^t is the set of agents from A that are currently working on task b_i at time unit t. An agent can only work on one task at a time, so $n_i^t \subseteq A$ and $\forall i \neq j, n_i^t \cap n_j^t = \emptyset$. All agents and tasks have a spatial location in the environment and the travel time, $TT(x, y)$, between two locations, x and y, is assumed to be computable. The travel time causes a delay between the time an agent is active in one task and the time it can become active in another. This means $\sum_i n_i^t \leq |A|$ as some agents might be in transit.

Each agent provides work amount w per time unit once the agent has reached a task. Every task $b_i \in B$ has a cost defined with the following recursive relationship:

$$f_i^{t+1} = f_i^t + h_i(f_i^t) - w \cdot |n_i^t|, \tag{1}$$

where f_i^t starts at some initial cost f_i^0 and h_i is a positive semidefinite monotonically increasing function as per the model proposed in [10]. We assume the growth function, h_i, is known or reasonably estimated.

If at some time t the cost of task b_i, namely f_i^t, reaches or passes zero, the task is considered complete. For this reason, when f_i^t is non-positive, $h_i(f_i^t)$ is assumed to be zero and we do not allow agents to be assigned to this task. When $h_i(f_i^t) > w \cdot |n_i^t|$ this means f_i^t is strictly monotonically increasing over t, which means the task is growing faster than the assigned agents can reduce it. If this is true, the task will never be completed unless more agents are assigned to it at a later point. We assume all tasks can initially be completed, as the goal is unclear if tasks grow indefinitely. The time when the last task is completed is defined as t_s. The objective of our problem is to minimize the accumulated growth of all tasks, which we will denote by $R_{t_s} = \sum_t \sum_i h(f_i^t)$. The objective corresponds to reducing the negative effects of the tasks, for instance, minimizing the number of trees burnt before a forest fire is extinguished.

4 Max-Sum Formulation

A Distributed Constraint Optimization Problem can be formally defined as a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{A} = \{a_1, a_2, ...a_{|\mathcal{A}|}\}$ is a set of agents, $\mathcal{X} = \{x_1, x_2, ...x_{|\mathcal{X}|}\}$ is a set variables, $\mathcal{D} = \{D_1, D_2, ...D_{|\mathcal{X}|}\}$ are the domains for the variables in $\mathcal{X}$, and $\mathcal{C} = \{c_1, c_2, ...c_{|\mathcal{C}|}\}$ are utility functions. Each utility function, c_i, gives a value based on its connected variables, $\{x_{i_1}, x_{i_2}, ...\}$, specifically $c_i : D_{i_1} \times D_{i_2} \times ... \rightarrow \Re$. Approaches to solve DCOPs range from optimal techniques [9] to heuristics [3]. Optimal solutions require an exponential coordination overhead. Heuristic approaches provide no guarantees to solution quality, but have a low coordination overhead. As our model is an approximation of an unknown process which involves a potentially large number of agents, we use heuristic methods. Using an optimal technique would only solve for the approximate model and not guarantee an actual optimal solution. Specifically, we use binary max-sum in conjunction with tractable higher order potentials to more efficiently pass messages [15]. This can reduce the message passing complexity from $O(|B|^{|A|})$ to $O(|A| \log |A|)$. The use of binary variables means $D_i \in \{0, 1\}$.

Our model of max-sum is very similar to [12], as they use binary max-sum in RoboCup Rescue as well. However, in [12] the solution is one-shot using heuristics on the current situation rather than our approach which models the problem and anticipates and responds to future growth. Since we model an unknown growth rate, agents benefit by adjusting to noise or inaccuracies by using a modification of max-sum (see

Section 4.1). The heuristics in [12] are domain specific to RoboCup Rescue by assigning explicit values to tasks (i.e., fires) based on the intensity of the fire and the number of other agents assigned to the fire. Our max-sum model assumes task size follows the recursive relationship defined in Eq. 1, then computes the exact effect on the tasks. This allows our max-sum to generalize to any tasks whose size can be well approximated by the general recursive relationship in Eq. 1.

Next we discuss the specifics of our max-sum problem formulation. We treat the cost, f_i^t, as an algebraic variable that is dependent on the time t and number of static agents active $|n_i^t|$. The goal is to minimize R_{t_s}, the total amount of growth before all tasks are completed. Our task growth approximation for the model in Eq. 1 is $h(f_i^t) = g_i \cdot f_i^t$, where g_i is a constant found empirically. For ease of calculation, we assume that time is continuous rather than discrete so this means $\frac{\delta f_i^t}{\delta t} = g_i \cdot f_i^t$. Solving for f_i^t yields the well known exponential function: $f_i^t = D \cdot e^{g_i \cdot t}$, where D is the integration constant. The initial task cost, f_i^0, is used to determine an appropriate D for each task. The cost reduction from agents working on a task is represented as $\frac{\delta f_i^t}{\delta t} = g_i \cdot f_i^t - |n_i^t| \cdot w$, which can again be solved for f_i^t:

$$f_i^t = \frac{|n_i^t| \cdot w}{g_i} + D \cdot e^{g_i \cdot t}, \tag{2}$$

where again D is an integration constant, which is negative if the agents are completing the task faster than it is growing, namely $D < 0$ if and only if $|n_i^t| \cdot w > g_i \cdot f_i^t$. If the number of agents changes, then D will need to be recomputed. We can use further algebra to get the following equations:

$$f_i^t = \frac{|n_i^t| \cdot w}{g_i} + \left(f_i^0 - \frac{|n_i^t| \cdot w}{g_i} \right) \cdot e^{g_i \cdot t} \tag{3}$$

$$-R(|n_i^t|) = \frac{|n_i^t| \cdot w}{g_i} \cdot (\ln(\alpha) + 1 - \alpha) + f_i^t \cdot (\alpha - 1), \text{ with } \alpha = \frac{|n_i^t| \cdot w}{|n_i^t| \cdot w - f_i^t \cdot g_i} \tag{4}$$

Eq. 3 estimates the cost of task b_i at time t and an active amount of agents assigned $|n_i^t|$. Eq. 4 is the cost which will be added to the system from time t until task i is finished, which means $\sum_i -R(|n_i^t|) = -(R_{t_s} - R_t)$ as defined in Section 3. This equation is not well defined if $|n_i^t| \cdot w < f_i^t \cdot g_i$, so we evaluate it as to ∞.

The factor graph [6] used to solve this problem is shown in Figure 1, with circles as variables and squares as functions. Each agent a_i controls all $|B|$ binary task assignment indicator variables $v_{i,j}$, where j indicates the agent's assigned task, b_j. The constraint q_i ensures an agent is assigned to exactly one task, specifically:

$$q_i(v_{i,1}, v_{i,2}, \ldots v_{i,|B|}) = \begin{cases} 0 & \sum_j v_{i,j} = 1 \\ -\infty & \text{otherwise} \end{cases}. \tag{5}$$

s_j is the accumulated cost growth until the task is finished. Since we want to minimize the accumulated growth, the negative of this value is taken:

$$s_j(v_{1,j}, v_{2,j}, \ldots v_{|A|,j}) = -R(\sum_i v_{i,j}), \text{ with } R \text{ as in Eq. 4.} \tag{6}$$

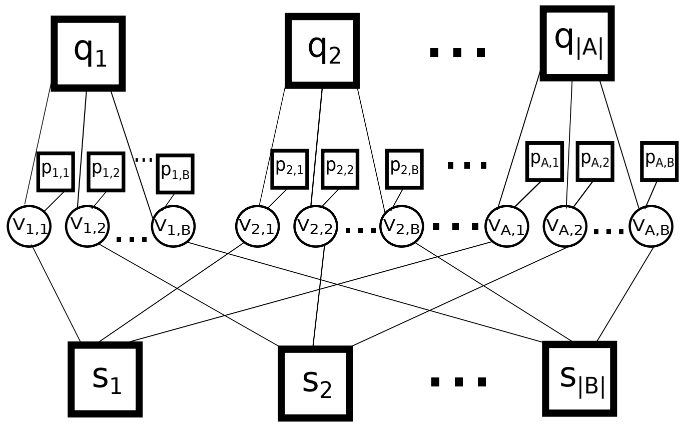


Fig. 1: The factor graph for our representation, with v as binary indicators of an agent's assignment, p as distance penalties, q as constraints for ensuring an agent is active on one task, and s as the utility function for the assignment detailed in Eq. 4.

$p_{i,j}(v_{i,j})$ is a travel penalty since s_j assumes the agent arrives instantly, so the correct amount is subtracted if $v_{i,j}$ is the only agent changing to this task. $TT(v_{i,j}, b_j)$ denotes the travel time between agent a_i and task b_j. The effect of the agent joining the task depends on how many other agents are already present. Knowledge of $\hat{T}$, the time it takes the currently assigned agents to complete the task, is approximated by the previous assignment to find $p_{i,j}(v_{i,j})$ as follows:

$$p_{i,j}(v_{i,j}) = \begin{cases} \frac{w \cdot e^{g \cdot \hat{T}}}{g} \cdot \left(e^{-g \cdot TT(v_{i,j}, b_j)} - 1\right) & v_{i,j} = 1 \\ 0 & v_{i,j} = 0 \end{cases} \tag{7}$$

This factor graph is binary and composed of only Tractable Higher Order Potentials, specifically only the cardinality of the inputs is required for s_j (number of agents assigned to the task) and q_i (number of tasks assigned to an agent) [15].

Our global utility function is a sum over all functions:

$$\sum_i q_i(v_{i,1}, v_{i,2}, ...) + \sum_i \sum_j p_{i,j}(v_{i,j}) + \sum_j s_j(v_{1,j}, v_{2,j} ...)$$

This factor graph is cyclical, so we cannot guarantee finding a global optimum. The global utility of the solution will tend to rise and then oscillate. To get the best quality solution, we keep a record of the highest utility solution ever seen. In addition, after max-sum ceases passing messages, we greedily assign agents one at a time to tasks as to maximize the global utility, and this assignment is used if it has higher utility.

Max-sum is run at every time step for multiple reasons. New tasks may appear, new agents may join and current agents may leave. Also as both the modeling and solution are approximate, readjusting projections with the real outcomes improves accuracy. Next, we add stability in the utility when faced with uncertainty in the environment.

4.1 Noise Resistant Max-Sum

Max-sum attempts to optimize the global utility and will change many assignments for a small gain. This works well if the utility evaluations are accurate, but as we will detail in Section 5, this causes much assignment thrashing when there is noise or the utilities are approximations. To create more stable assignments, we will incorporate uncertainty into our model, which enables us to balance risk vs. benefit. We call this the "Noise Resistant Max-Sum" (NR Max-Sum).

The inaccuracies that max-sum must handle stem from a variety of sources, such as the regression model, error in the empirically evaluated parameters, and noise in the tasks growth. We use the recurrence relation in Eq. 1, which has a few factors which need to be derived. We assume that only the cost f_i^t and assignment n_i^t are known exactly. This means we must fit h_i and w to the domain of interest.

Both of these parts are integral in the shape of the function and the assignment of agents. This means we cannot compartmentalize the uncertainty to a specific section of our problem. For this reason, we apply the uncertainty directly into Eq. 1 and decide if the assignments derived are sufficiently better than the previous assignments. Max-sum is chosen to derive new assignments as it has been shown to yield good results in a decentralized manner, but this noise resistance can be applied to any distributed coordination algorithm. Specifically, we run max-sum as normal until it yields an assignment, and then recompute the global utility by modifying Eq. 1 as $f_i^{t+1} = f_i^t + h_i(f_i^t) - w \cdot |n_i^t| + \mathcal{N}(0, \sigma_{h_i}^2 + \sigma_w^2)$. Here $\sigma_{h_i}^2$ and σ_w^2 are the estimated variances of h_i and w respectively.

We then sample the normal distribution and compare the resulting effect on both the previous assignment and the newly computed assignment. If the new assignment is better both on average and in a specific percent of the samples drawn, then the new assignment is adopted otherwise the old assignment is kept. The expected utility ensures that on average there should be a gain in the new assignment and the superior performance ratio reduce the effect of outlier samples and helps ensure the assignments are stable. After parameter tuning, we found that requiring 70% of the samples to be better works well, but if solution outliers (both good and bad) are less of an issue then this percent could be lowered. This parameter depends on the accuracy of the model: the greater the noise or inaccuracy the higher this parameter should be.

5 Results

We compare our noise resistant max-sum (NR Max-Sum) against the modified version of RT-LFF based on LFF described in [10]. LFF is a one-shot algorithm that maximizes the amount of time agents work on tasks. This provides an optimal solution in a few cases, but often the inefficiency of assignments causes this algorithm to perform suboptimally. RT-LFF is a real-time modification of LFF that can react to noise and inaccurate modeling. RT-LFF is a centralized conservative method that only changes assignments if a large gain is detected. We modify RT-LFF which originally attempted to minimize finish time of the last task to instead minimize the accumulated task growth to correspond with our problem definition. This modified RT-LFF has a worst-case running time of $O(|A||B|^2)$, but is typically significantly less in practice.

NR Max-Sum, max-sum and RT-LFF are compared in both a controlled simple simulator and in the RMASBench [5] extension of RoboCup Rescue to show the applicability to complex environments and to make our results easily comparable. We provide the optimal solution, which is computed assuming the travel time between tasks is zero so it gives an unreachable lower bound when distance takes time to traverse.

5.1 Simple Simulator

The simple simulator gives us greater analytical power than RoboCup Rescue since every aspect can be controlled. We can explicitly define the initial cost, f_i^0, growth functions, h_i, and even the effect of agents on the task, w, for any number of tasks. Travel time between tasks can be asymmetric, for example if a task is on top of a hill it will be quicker to descend than ascend.

We consider the optimal, RT-LFF, max-sum and NR Max-Sum in different configurations to highlight their strengths and weaknesses. First we compare these algorithms for different types of growth fucntions (Table 1), then we analyze the performance of these algorithms under noise and modeling inaccuracies (Table 2). All the experiments in this section use 200 agents with work rate $w = 0.0015$. The number of agents is large to minimize discretization effects. Coefficients for w and in the growth functions are small to minimize the discretization of the time steps.

Table 1 compares the algorithms across four different sets of growth functions, h_i, each with very different optimal solutions. Each entry is the average over five runs, with the NR Max-Sum assuming a 10% error in growth function despite there being none. The optimal solution requires no travel time between tasks, $TT(x, y) = 0$, so this setting is used for all algorithms in this table. The "mixed" row uses a different h_i function for each task, respectively $h_1(x) = x^2$, $h_2(x) = x$, and $h_3(x) = \sqrt{x}$. There is no known optimal solution for this case when the growth functions are different.

We see that max-sum and NR Max-Sum perform quite close to optimal in all cases. RT-LFF compares well, except for the concave growth function, namely $\sqrt{x}$. This is because RT-LFF attempts to minimize the number of agents switched, but the optimal solution for this type of task is to have all agents on the same task and move them around together. As no initial settings are changed between runs, only the probabilistic NR Max-Sum yields different results between trials. Using ANOVA with repeated measures gives a p-value of 0.1698, which indicates none of the methods performs any significantly different than each other, including the optimal.

When h_i is approximated or noisy, the differences between the algorithms are more distinctive. The average of five runs is shown in Table 2 using for cost growth $h_1(x) =$

Table 1: Accumulated growth cost, R_{t_s} for different setups in the simple simulator

Growth functions, $h_i(x)$	Initial task costs, f_i^0	Optimal	RT-LFF	Max-Sum	NR Max-Sum
$0.00019 \cdot x^2$	{10, 25, 30}	104.39	128.25	104.39	104.50
$0.00360 \cdot x$	{10, 30, 40}	188.72	188.72	188.72	188.72
$0.02000 \cdot \sqrt{x}$	{5, 10, 15, 20}	43.72	136.70	43.75	43.75
$mixed$	{25, 10, 10}		31.25	24.48	24.48

Table 2: Accumulated growth cost, R_{t_s} in the simple simulator experiments

	Optimal		RT-LFF		Max-Sum		NR Max-Sum	
	μ	σ	μ	σ	μ	σ	μ	σ
$known$	30.71	0	36.23	0	33.50	0	33.53	0.02
$known + noise$	33.37	2.26	41.05	0.50	42.48	1.55	37.51	0.72
$unknown$	14.06	0	15.04	0	16.75	0	14.61	0.19
$unknown + noise$	15.46	1.38	23.82	0.29	30.50	1.71	21.19	0.72

$h_2(x) = 0.00019{\cdot}x^2$, initial task sizes $f_1^0 = 20$, $f_2^0 = 30$, and travel time $TT(x, y) = 5$, When noise is added, the growth function is changed to $\hat{h}_i(x) = h_i(x) + \mathcal{N}(0, 0.02)$. If the function is unknown the methods assume $h(x)$ is defined as above, but the simulator instead uses $H_1(x) = H_2(x) = 0.000006 \cdot x^3$ and $\hat{H}_i(x) = H(x)_i + \mathcal{N}(0, 0.02)$ for the cases without and with noise respectively. The optimal solution benefits over the other algorithms in that it always knows the true growth function h_i, agents have no travel time between tasks and can assign based on the randomly generated noise. These add up to a large advantage, but ensures that the true optimal is found.

From Table 2, we see that NR Max-Sum minimizes R_{t_s} more than the other algorithms, with the exception of the optimal. Max-sum performs moderately on both known and unknown task growth rates, but does quite poorly when noise is introduced. This is because max-sum tries to adapt the assignments to changes even if they are caused purely by noise. With a travel time of 5, max-sum typically has two agents switching back and forth due to the random noise. While this is only one percent of the agents, it is wasteful and illogical to constantly send agents back and forth between two tasks not doing any work. The NR Max-Sum thrashes much less because the built in uncertainty makes it difficult for the model uncertainty to surpass 70% of the samples. Yet, if there is a definite gain it is able to capitalize on it.

RT-LFF has a fairly conservative heuristic and thus rarely thrashes. However, this also means it bypasses many opportunities for small advantages and only takes them if they become large. For this reason, RT-LFF did not perform as well as max-sum or NR Max-Sum when the cost growth function was known. The very stable assignments also reduce the average travel time of agents considerably in this method.

Treating the noise levels as a blocking factor, we see that an ANOVA test gives a p-value of 0.01668. Using the Tukey HSD test, we find that the $\{$Optimal, max-sum$\}$ pair is statistically significant (p-value = 0.0142297) and the $\{$Optimal, RT-LFF$\}$ pair is close (p-value = 0.0584526). This shows that NR max-sum does not perform significantly worse than the optimal solution for the tested function. In the simple simulation the NR max-sum was not shown to be conclusively better than RT-LFF or max-sum, which we will instead show in the RoboCup Rescue Simulator next.

5.2 RoboCup Rescue Simulator

In this section we focus on the problem of dealing with fires in the RoboCup Rescue simulator. The RoboCup Rescue simulator is designed for urban search and rescue after an earthquake, where buildings collapse and fires start in buildings. The environment is complex with thousands of buildings and hundreds of agents in the full simulation extracted from street maps of real cities. The full simulator uses heterogeneous agents, but for this work we focus only on the agents that can extinguish fires, i.e. firetrucks, through the use of the RMASBench simulator extension [5]. Although RMASBench is an extension of the RoboCup Rescue Agent Simulator, the communication is relaxed to allow many more messages to be sent than in the original simulator, along with multiple messages per time step.

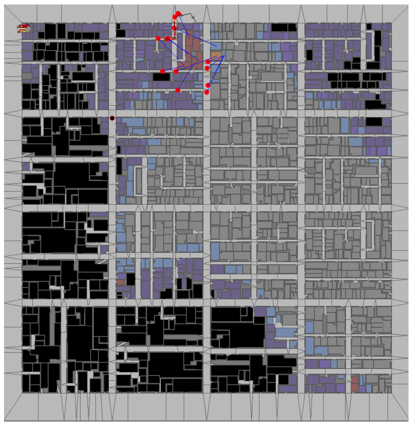
Fires are the most dangerous hazard in RoboCup Rescue. Buildings heat up and catch on fire based on how many other nearby buildings are on fire. This creates a positive feedback loop, which causes fires to grow at exponential rates. A screenshot of the simulator is shown in Figures 2a and 2b. Red dots represent fire trucks, dark gray polygons are buildings while light gray polygons are roads. Buildings on fire are yellow, orange and red in increasing intensity and temperature. If a building burns too long, it will become completely destroyed and turn black. When a fire truck extinguishes a building, it will become blue or purple. In the RoboCup Rescue simulator, fires are defined per building. Nearby fires are clustered into a single task, where the cost is the number of buildings on fire in the cluster.

This clustering is done using bottom-up hierarchical clustering with the Euclidean distance as the metric and the minimum distance between all pairs linkage criteria. If the distance between the closest pair of clusters is over 50 meters in the simulation, then the clustering would cease.

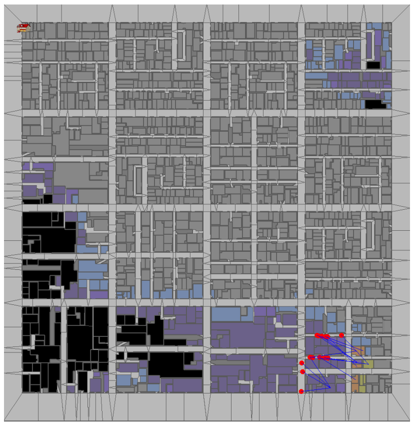
Now that tasks are well defined for the problem, h_i and w also need to be estimated. As described earlier in Section 4, we will approximate the cost growth function as for all tasks as $\forall i\ h_i(f_i^t) = g \cdot f_i^t - |n_i^t| \cdot w$. To estimate g, buildings were allowed to burn unhindered for 100 simulation steps in 20 different tests. When using exponential regression to find the best fit of the data, we have the choice of modeling this as $c \cdot e^{d \cdot t}$ or simply $e^{d \cdot t}$, where d is the g we want. The two regression models that minimize the sum of squares are: $4.6852 \cdot e^{0.0393 \cdot t}$ or $e^{0.0561 \cdot t}$. We cannot directly use the first regression model as h_i is not defined in this way, so we will simply drop the scaling factor and consider two possible g values: 0.0393 and 0.0561.

The work rate, w, was also empirically derived. A fixed small number of fires were repeatedly extinguished in 70 tests. If f_i^0, g, $|n_i^t|$ and f_i^t for some t are known, Eq. 3 can then be used to solve for w. The experiments found that w is rather noisy and typically has a few high outliers. This means in some cases, the fire trucks were able to extinguish the fire much more quickly than the model anticipated. The lack of low outliers is ideal and provides stability, since this indicates agents do not extinguish poorly very often. When $g = 0.0393$, the variance in w is decreased in addition to having fewer outliers. We decided to use this value for g since it stabilizes w more. In the end our model for projecting cost depends on both g and w, so it makes sense to optimize them together.

Virtual City 4 (VC 4) screenshots of max-sum and NR Max-Sum are shown in Figures 2a and 2b respectively. Max-sum is overly responsive to noise and model inaccuracies, so when a few more buildings unexpectedly start on fire or get extinguished, agents will likely transfer. This can cause severe assignment thrashing for max-sum, while NR Max-Sum is less responsive as a threshold must be reached before assignments are changed. Assignment thrashing for max-sum and not for NR Max-Sum is

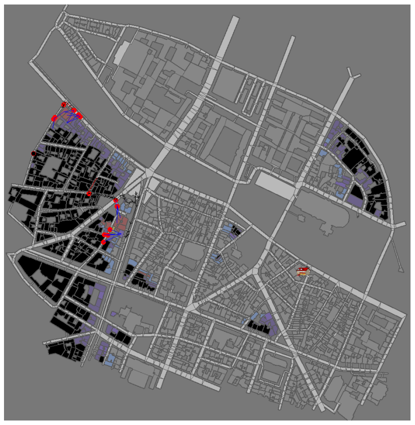


(a) Max-sum near the end with 42% of the buildings intact

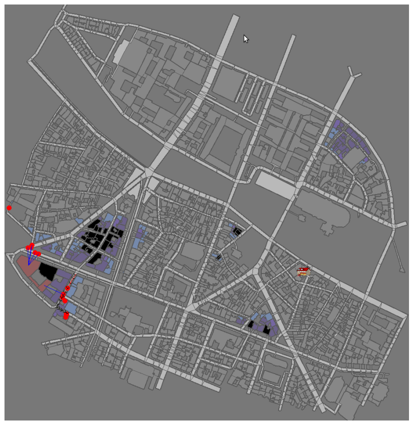


(b) NR Max-Sum near the end with 76% of the buildings intact

Fig. 2: Virtual City 4 (VC 4) map



(a) Max-sum near the end with 71% of the buildings intact



(b) NR Max-Sum near the end with 87% of the buildings intact

Fig. 3: Paris 1 map

also clear in Paris 1, shown in Figures 3a and 3b respectively. Differences as large as shown are hard to create in maps and not seen on all maps [11], but our results show NR Max-Sum performing better on average.

Table 3 shows the percentage of intact buildings for 5 variations of both the Virtual City (VC) and Paris map. Each variation is denoted by a number after the map name and changes the number and location of fires along with the number and location of agents. Virtual City is a smaller map, so the number of agents and fires range between 8 to 15 and 30 to 50 respectively. In Paris there are between 10 to 50 agents with 50

Table 3: Percent of buildings intact at the end of simulation on different versions of Virtual City (VC) and Paris maps

Map	RT-LFF		Max-Sum		NR Max-Sum	
	μ	σ	μ	σ	μ	σ
VC 1	27.86	7.23	16.18	6.03	35.92	5.58
VC 2	81.04	0.43	78.98	0.33	81.66	0.49
VC 3	49.59	9.70	73.81	6.36	73.02	4.26
VC 4	70.98	2.90	43.47	4.50	74.84	3.94
VC 5	23.10	2.69	17.58	1.34	23.64	1.13
Paris 1	84.53	4.55	72.54	5.03	88.98	5.11
Paris 2	41.11	7.00	31.37	8.73	44.34	9.06
Paris 3	60.73	4.69	68.81	3.92	69.12	3.56
Paris 4	91.24	0.45	90.99	0.52	91.97	0.41
Paris 5	74.78	6.36	79.64	5.66	85.64	3.62

to 120 fires. This randomization causes some configurations to be easier to solve and score higher than others. Each configuration was run 5 times. We compared the same algorithms used in Section 5.1, with the exception of the optimal algorithm. Since h_i and w are only approximations, we cannot compute the optimal solution in this case. Some results, such as VC 5, are poor across all algorithms. This indicates more that VC 5 is a hard configuration, rather than the efficiency of the algorithms. The opposite is the case in VC 2, where the configuration is too easy and all algorithms scored well.

There is a large difference in score between agents barely containing fires and fires slowly spreading, which cause algorithms to have bimodal score distributions [11]. Also as mentioned above inherent map difficulty can cause algorithms to score similarly. For these reasons we apply the non-parametric Wilcoxon signed-rank test instead of the normal t-test. As NR Max-Sum performs on better on all maps in comparison to RT-LFF, the test statistic $z = 2.7775$ which is greater than the $\alpha = 0.05$ critical level of 1.960. Virtual City 3 is the only map where max-sum outperforms NR Max-Sum ($z = 2.5734$), which implies NR Max-Sum scores significantly higher than both of these methods.

6 Conclusions and Future Work

In this work we focused on solving tasks with costs that grow over time. We provide the first decentralized task assignment approach that considers task with growing cost over time. Our solution, which is based on the max-sum algorithm, outperforms the previous centralized solution in both a simple simulation and the RoboCup Rescue Agent Simulator.

Our functions that predict the future cost of tasks, assume that the number of agents active on a task is constant. However, many times it is beneficial to reassign agents before the end of the simulation. This causes our predictions of task cost to be inaccurate since they assume static agent assignments. Future work will consider the effect of planned reassignments over time. The current work also assumes that these functions are estimated upfront, but they could be learned over the course of the simulator. Given different families of functions, the model could be refined over time to select the best function family and parameters. This could also help determine what errors come from modeling versus inherent randomness of the growth.

Acknowledgments: Work supported in part by the Safety, Security, and Rescue Research Center at the University of Minnesota (NSF IIP-0934327).

References

1. Amador, S., Okamoto, S., Zivan, R.: Dynamic multi-agent task allocation with spatial and temporal constraints. In: Proc. Nat'l Conf. on Artificial Intelligence. pp. 1384–1390 (2014)
2. Atlas, J., Decker, K.: Coordination for uncertain outcomes using distributed neighbor exchange. In: Int'l Conf. on Autonomous Agents and Multi-Agent Systems. pp. 1047–1054 (2010)
3. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: Int'l Conf. on Autonomous Agents and Multi-Agent Systems. pp. 639–646 (May 2008)
4. Kitano, H., Tadokoro, S.: RoboCup Rescue: A grand challenge for multiagent and intelligent systems. AI Magazine 22(1), 39–52 (2001)
5. Kleiner, A., Farinelli, A., Ramchurn, S., Shi, B., Maffioletti, F., , Reffato, R.: RMASBench: Benchmarking dynamic multi-agent coordination in urban search and rescue. In: Int'l Conf. on Autonomous Agents and Multi-Agent Systems. pp. 1195–1196 (2013)
6. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. Information Theory, IEEE Transactions on 47(2), 498–519 (Feb 2001)
7. Léauté, T., Faltings, B.: Distributed constraint optimization under stochastic uncertainty. In: Proc. Nat'l Conf. on Artificial Intelligence. pp. 68–73 (2011)
8. Melvin, J., Keskinocak, P., Koenig, S., Tovey, C., Ozkaya, B.: Multi-robot routing with rewards and disjoint time windows. In: Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems. pp. 2332–2337 (Oct 2007)
9. Modi, P.J., Shen, W.M., Tambe, M., Yokoo, M.: ADOPT: asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence Journal 161, 149–180 (2005)
10. Parker, J., Gini, M.: Tasks with cost growing over time and agent reallocation delays. In: Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems. pp. 381–388 (2014)
11. Parker, J., Godoy, J., Groves, W., Gini, M.: Issues with methods for scoring competitors in RoboCup Rescue. In: Autonomous Robots and Multirobot Systems at AAMAS (2014)
12. Pujol-Gonzalez, M., Cerquides, J., Farinelli, A., Meseguer, P., Rodriguez-Aguilar, J.A.: Efficient inter-team task allocation in RoboCup Rescue. In: Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems. pp. 413–422 (2015)
13. Ramchurn, S., Farinelli, A., Macarthur, K., Polukarov, M., Jennings, N.: Decentralised coordination in RoboCup Rescue. The Computer Journal 53(9), 1–15 (2010)
14. Stranders, R., Tran-Thanh, L., Fave, F.M.D., Rogers, A., Jennings, N.R.: DCOPs and bandits: exploration and exploitation in decentralised coordination. In: Int'l Conf. on Autonomous Agents and Multi-Agent Systems. pp. 289–296 (2012)
15. Tarlow, D., Givoni, I.E., Zemel, R.S.: Hop-map: Efficient message passing with high order potentials. In: Proc. Int'l Conf. on Artificial Intelligence and Statistics. pp. 812–819 (2010)
16. Toth, P., Vigo, D. (eds.): The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA (2002)