Exploiting Social Context for Anticipatory Analysis of Human Movement (Final Report)

Dr. Gita Sukthankar
University of Central Florida
Department of EECS
4000 Central Florida Blvd
Orlando, FL, 32816-2362
Email: gitars@eecs.ucf.edu
Tel: 407-823-4305

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
OCT 2012		

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Exploiting Social Context for Anticipatory Analysis of Human Movement	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
University of Central Florida,Department of EECS,4000 Central Florida Blvd,Orlando,FL,32816	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**33**	

Exploiting Social Context for Anticipatory Analysis of Human Movement (Final Report)

Dr. Gita Sukthankar
University of Central Florida
Department of EECS
4000 Central Florida Blvd
Orlando, FL, 32816-2362
Email: gitars@eecs.ucf.edu
Tel: 407-823-4305

SUMMARY

Our aim is to develop principled methods to transfer models of human movement using social context. The resulting techniques will form a fundamental contribution to the field of human terrain analysis, enabling diverse sources of data to be leveraged along with GEOINT and resulting in improvements to software tools used by analysts for the anticipatory analysis of human behavior. The philosophy behind our proposed approach is the following:

- it is more effective to represent an agent's social context with specific exemplars of people who share socio-cultural similarities than it is to create a parametric model over the entire population
- biased sampling techniques can allow the leveraging of large collections of data from groups of humans without necessitating the creation of an explicit model of interpersonal interaction effects
- existing human behavior models are best used to supplement data gaps.

Our goal is to reduce the analysts' workload by identifying the relevant regions and time-frames in spatio-temporal data sets. This information can be used: (1) create intelligent data filters, (2) guide the future deployment of data collection capabilities, and (3) assess competing hypotheses. The information extracted using our techniques (augmented social networks, points of interests, reduced road networks) can be visualized and modified by the analyst to modify the search boundaries in an interactive fashion.

PROGRAMMATIC

- Attended and presented update at Panel 6: Oct 6-8, 2010
- Attended NGA summit Mar 15-16th, 2012

- **PUBLICATIONS**
  - **Journal article** Journal of Pervasive and Mobile Computing (B. Tastan and G. Sukthankar, *Leveraging Human Behavior Models to Improve Path Prediction and Tracking in Indoor Environments, 2011* vol. 7, pp. 319--330.)
  - **Conference paper at the International Conference on Pattern Recognition (ICPR 2012)** (L. Zhao, G. Sukthankar, and R. Sukthankar, *Importance-Weighted Label Prediction for Active Learning with Noisy Annotations*)
  - **Conference paper at the International Conference on Social Computing (SocialCom 2011)** (L. Zhao, G. Sukthankar, and R. Sukthankar, *Incremental Relabeling for Active Learning with Noisy Crowdsourced Annotations)*

- **Conference paper at International Conference on AI and Interactive Entertainment (AIIDE 2011),** (B. Tastan and G. Sukthankar, *Learning Policies for First Person Shooter Games by Demonstration)*
- **Conference paper at International Joint Conference on AI (IJCAI 2011),** (K. Laviers and G. Sukthankar, *A Real-Time Opponent Modeling System for Rush Football)*
- **Best poster for extended abstract at the Conference of the Florida AI Research Society (FLAIRS 2012)** (B. Tastan, D. Chang, and G. Sukthankar, *Learning Motion Prediction Models for Opponent Modeling)*
- **Extended abstract at the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2012)** (B. Lewis and G. Sukthankar, *Configurable Human-Robot Interaction for Multi- Robot Manipulation Tasks)*
- **Extended abstract at CompleNet (Workshop on Complex Networks 2012)** (M. Maghami and G. Sukthankar, *Influence Maximization for Advertising in Multi-agent Markets)*
- **Extended abstract at Human-Agent Robot Teams workshop** on *Imbuing Human-Robot Teams with Mutual Predictability* (Dec 13-17, 2010 Leyden, Netherlands)

FUNDING

- **Cumulative incurred expenses: (how much of the obligated funding)** 95%

MILESTONES ACHIEVED

Task 1: Extracting Social Context from Human Terrain. Description: novel algorithm for obtaining context from social networks and similar agents from general population.

- **Agent Movement Simulator**
  http://code.google.com/p/agent-movement-simulator/

This simulator generates the schedule of a custom number of agents in a social network and their social interactions with other agents. The dataset produced from this project can mimic a real geo-social dataset. We present a machine learning approach for modeling the user's social context and incorporating it into a destination prediction system. Subtle correlations between the transportation patterns of two classes of user associates, neighbors who are spatially co-located, and friends who share a social connection, are exploited by our user model to improve the accuracy of a set of classifiers trained using Adaboost to recognize destinations from partial trajectories (see attached papers).

Task 2: Inferring Agent Preferences from Observed Trajectories. Deliverable: Novel algorithm for trajectory prediction and transfer to unseen environments.

- **Human Steering Model and Particle Filter Tracker**
  https://github.com/bulenttastan/IAL-ParticleFilter_HSM

To track humans with sensor networks, detect behavior anomalies, and offer effective navigational assistance, we need to be able to predict the trajectory that a human will follow in an environment. Although human paths can be approximated by a minimal distance metric, humans often exhibit counter-intuitive behaviors; for instance, human paths can be non-symmetric and depend on the direction of path traversal (e.g., humans

walking one route and returning via a different one). To address this problem, a psychologically-grounded model of human steering and obstacle behavior are incorporated into the tracking and goal prediction system.

Task 4: Learning through Social Context. Deliverable: Novel algorithm for transfer learning. Crowdsourcing has become a popular approach for annotating the large quantities of data required to train machine learning algorithms. However, obtaining labels in this manner poses two important challenges. First, naively labeling all of the data can be prohibitively expensive. Second, a significant fraction of the annotations can be incorrect due to carelessness or limited domain expertise of crowdsourced workers. Active learning provides a natural formulation to address the former issue by affordably selecting an appropriate subset of instances to label. Unfortunately, most active learning strategies are myopic and sensitive to label noise, which leads to poorly trained classifiers. We developed an active learning method that is specifically designed to be robust to such noise.

Task 5: Anticipatory Analysis of Human Behavior. Deliverable: final report and software
UCF Agent Based Modeling Transportation Simulation:
http://code.google.com/p/ucf-abm/
An activity-based microsimulation model for transportation, dining, parking, and building occupation preferences on the UCF campus (see attached paper).

OUTCOMES
1) Created methods for both short and long-term prediction of human transportation patterns and validated them on a variety of human datasets.
2) Developed a model for how social context can affect transportation patterns and validated it on a simulated dataset.
3) Collected survey data on student transportation patterns on the UCF campus and created an activity-based microsimulation of campus activity.
Summary: Due to their cheap development costs and ease of deployment, surveys and questionnaires are useful tools for gathering information about the activity patterns of a large group and can serve as a valuable supplement to tracking studies done with mobile devices. However in raw form, general survey data is not necessarily useful for answering predictive questions about the behavior of a large social system. We developed a method for generating agent activity profiles from survey data for an agent-based model (ABM) of transportation patterns of 47,000 students on a university campus. We compare the performance of our agent-based model against a Markov Chain Monte Carlo (MCMC) simulation based directly on the distributions fitted from the survey data. A comparison of our simulation results against an independently collected dataset reveals that our ABM can be used to accurately forecast parking behavior over the semester and is significantly more accurate than the MCMC estimator.
4) Developed techniques for using crowdsourcing to replace the survey data-gathering process. This process of crowdsourcing geo-tagged data will be the focus of our Phase 3 efforts.

Exploiting Social Context for Destination Prediction

Erfan Davami
Department of EECS
University of Central Florida
Orlando, FL US
Email: e.davami@gmail.com

Gita Sukthankar
Department of EECS
University of Central Florida
Orlando, FL US
Email: gitars@eecs.ucf.edu

***Abstract*—Early and accurate destination prediction is an important enabling technology for a variety of ubiquitous computing applications, including driver assistance systems and mobile phone apps. In this paper, we present a machine learning approach for modeling the user's *social context* and incorporating it into a destination prediction system. Subtle correlations between the transportation patterns of two classes of user associates, *neighbors* who are spatially co-located, and *friends* who share a social connection, are exploited by our user model to improve the accuracy of a set of classifiers trained using Adaboost to recognize destinations from partial trajectories. Our results serve as a pointer to designers of mobile applications on how to aggregate information across the user's social network to improve behavior prediction accuracy.**

***Keywords*-destination prediction; boosting; social networks;**

I. Introduction

In the course of a single day, humans make hundreds of minor decisions about their future actions—-where to eat, which route to take home, and what time to leave work. These choices are impacted by their social context; however in many cases this influence is subtle, not easily quantified, nor directly correlated with immediate events. The dual forces of homophily and social influence have been shown to engender both attitude and behavior similarities in social systems. For instance, Schelling models have been used to predict longer-term geographic effects of human neighbor preferences [1]. The aim of our research is to utilize the social context to predict short-term effects of one's associates on user transportation preferences. Models such as social potential fields [2] can be used to predict trajectories at short time scales but do not account for non-local influences. Recommendation systems [3] and reputation networks [4] attempt to quantify the impact of social context on a single choice (e.g., a book purchase or movie viewing) but are not designed for sequential decision-making problems. In this paper, we describe a technique for learning models of social context to improve the prediction of destinations from partial walking, driving, and biking trajectories.

II. Related Work

A variety of machine learning approaches have been applied to the problem of learning models for destination prediction. Krum and Horvitz [5] introduced a technique called predestination for learning a probabilistic map of destinations from the Microsoft Multiperson Location Survey (MSMLS) data. Hidden state estimation approaches such as dynamic Bayesian networks are a natural fit for the problem since the structure of the model is highly constrained by the road network [6]. For driver prediction, an important intermediate step is associating noisy GPS readings with the correct road network segments [7]. However, supervised learning approaches such as conditional random fields have shown good performance and are less constrained by the independence assumptions [8]. Human transportation patterns can also be modeled as the outcome of a rational process in which the user simply seeks to maximize reward and tackled with inverse reinforcement learning [9]. In this paper, we use a set of binary decision trees to classify partial trajectories, but the proposed social context features could easily be employed as part of a CRF or DBN model.

The idea that there are strong behavior correlations across related individuals has been explored within the Reality Mining dataset. Repetitive patterns in human behavior (eigenbehaviors) were extracted using principal component analysis and clustered together to identify group affiliations with a high degree of accuracy [10]. The authors propose that it is possible to their model to build demographic profiles to bootstrap new user models. More recently, Community Similarity Networks (CSN) have been proposed as a mechanism for explicitly utilizing inter-personal similarity to train individual user models for activity recognition [11]. Our proposed method uses transportation data from neighbors and friends to improve individual user models but does not require the explicit construction of other user or group models to improve prediction accuracy.

III. Methodology

Our proposed approach leverages the user's social and geographic connections to improve destination prediction. These special social context features are extracted from the dataset and used to augment the user model. Using boosting, an ensemble learning approach, a set of multi-class decision trees is trained using partial trajectories plus social context to predict the user's final destination. Our results conclusively

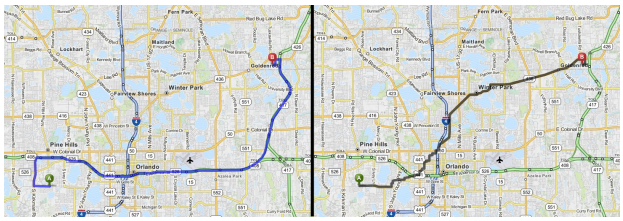


Figure 1. Trajectory of an agent while driving (left) or walking (right). Note that the trajectory of an agent depends on the transportation modality of the agent even if the start and end points of the path are identical.

show that the social context features improve the destination prediction accuracy of the user model, especially at the crucial early stages when there is relatively little trajectory information. Also, in absence of user location data, the social context alone can be used to predict the user's destination at later stages of the journey.

A. Social Simulation

To evaluate the approach, we developed an Android application in conjunction with an agent-based social simulation of a large urban area to generate GPS trajectories of 2000 users' movements over 28 days (Figure 1). One issue that we observed with our and other mobile device datasets is that they often are disproportionately drawn from one urban area. Also, recruiting groups of associates (friends and acquaintances) can be difficult, resulting in incomplete social context. Creating a simulated dataset has the advantage of significantly widening the range of agent behaviors and destinations, and avoiding privacy concerns while capturing social context information. Our dataset will be publicly released to enable direct comparisons.

Each agent represents a user with a distinct schedule, set of transportation preferences, and list of potential destinations including the user's house, work place, two shopping areas and four other locations sampled from other categories such as schools and entertainment centers. The agent has a clock governing its daily schedule and also maintains some continuity in its weekly schedule. From the time that the agent awakens until bedtime, it moves between destinations as dictated by its schedule; a Gaussian noise distribution is used to model variance in the agent's arrival, departure times, and duration of miscellaneous activities. Figure 2 shows the schedule of a typical agent (agent 281 on day 5) in the training set.

Eighty destinations (most within a 15-mile radius) in an urban area centered at latitude 28.53709, longitude -81.38179 were selected. Within our simulation, the transportation network is represented as a directed graph with eighty nodes and three edges connecting each node. The three edges correspond to the path taken by an agent; agents can use three different transportation modalities (driving, walking or cycling) to travel between destinations. Plausible

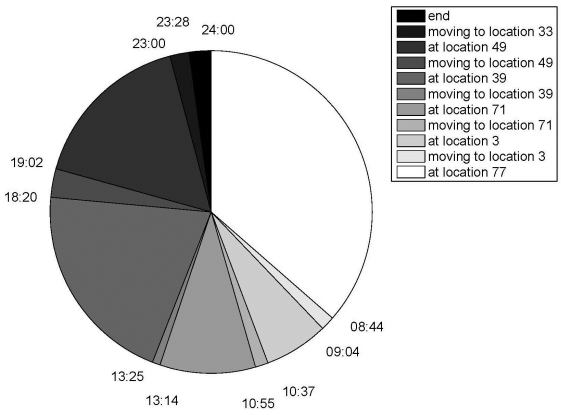


Figure 2. The schedule of agent 281 on day 5. The agent starts the day at 8:44am. At 9:04am it reaches its workplace (location number 3 in the map). It performs an errand at location 71 at 10:55am. Before returning home, the agent stops at location number 33 (shopping center) at 23:28.

paths between destinations are obtained through a free online tool called Mapquest Open Directions API Web Service [12]. Each path structure contains the path length, duration of travel, the list of the agent's maneuvers, and the time of each maneuver.

Using the path data and the schedule of each agent in each day, the trajectory set of that day can be constructed. Each data point contains the agent's geolocation information and its social context. The features include agent coordinates, time and day of observation, speed and direction of movement, and current coordinates of the top 5 friends and neighbors. Simulated GPS trajectory data from 21 days was used to train the destination prediction models that were evaluated with the remaining 7 days of agent observations.

B. Modeling Social Context

Each agent's social context is represented by the locations of two types of associates: *neighbors* and *friends*. However, we take a broad view of the concept of neighbor. Rather than defining an agent's neighbors as simply the set of agents whose homes lie within a certain radius of the agent's home, we use the set of agents that frequent areas that are generally close to the user's trajectory set. This captures the idea that there are often people who spend a large amount of time within the same territory and utilize the same schools, shopping areas, and miscellaneous destinations.

As the population of agents in the dataset grows, it becomes more likely that agents have overlapping transportation patterns. Hence the nearest neighbor of agent a is actually defined as the agent a' that minimizes the average distance between a and a' (at every time step) over the set of training trajectories. Naively computing the average distance between each pair of agents over the dataset is computationally expensive with worst case $O(N^2T)$ for 2000 agents and

total observations T=8,555,217. Fortunately, we are able to reformulate the problem as sketched below to obtain a more efficient approach:

$$\begin{aligned}
\frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{\sum_{t\in\mathcal{T}_d}X}{|\mathcal{T}_d|} &= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{\sum_{t\in\mathcal{T}_d}(I_d+C_t)}{|\mathcal{T}_d|} \\
&= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\left(I_d+\frac{\sum_{t\in\mathcal{T}_d}C_t}{|\mathcal{T}_d|}\right) \\
&= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}I_d+\frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{\sum_{t\in\mathcal{T}_d}C_t}{|\mathcal{T}_d|} \\
&= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}I_d+\frac{1}{|\mathcal{T}|}\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}_d}C_t \\
&= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}I_d+\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}C_t.
\end{aligned}$$

Here, X denotes the matrix capturing distances between agents at a particular instant in time; $\mathcal{D}$ are days in the training set ($|\mathcal{D}|$=21); $\mathcal{T}$ is the complete set of observations (over all days); and $\mathcal{T}_\lceil$ denotes the observations on day d; I_d denotes the initial distance matrix between agents at the start of day d and C_t is a change matrix at a given point in the observation.

The second aspect of social context is friendship. The best friend of agent a is simply defined as being the individual with the most encounters with a. An encounter is defined as a situation where two agents are in the same place at the same time and thus have the potential to meet. For example, if a is at location number 37 from 8:00am to 9:00am, and agent b is at that same location from 8:30am to 9:15am, there is spatio-temporal overlap and agents a and b record an encounter. The location in which an encounter takes place could be a movie theater, a bar, a library or even a shopping mall. Naturally there can be chance encounters between agents who are not actually friends but repeated encounters over a longer period of time are likely to denote some level of acquaintance of friendship. Thus the best friend relationship is calculated as:

$$\text{bestFriend}_a = \arg\max_{a'\in A-a}\sum_{D}\text{encounter}(a,a')$$

where D denotes the days over which training data was collected. The friendships between agents form a sparse matrix F in which element F_{ij} encodes the total number of encounters between agents i and j. The geographic location of each of the top five best friends form the second part of the agent's social context. Although assuming that emotional and geographic distance are directly correlated is a highly simplified model of human friendship, for the purposes of predicting transportation patterns it is most useful for our model to exclude long-distance relationships and overweight the importance of propinquity.

C. Learning the Model

A sequence of 10 data points from each trajectory are used to train the models and predict the destination of each individual agent. These sample points are selected at uniform time intervals from the start of the movement up to the agent's current location. The problem is framed as a multi-class classification task where the classifier makes a forced choice between one of the eighty destinations. The set of classifiers is trained with multi-class Adaboost classifier designed for this purpose.

Adaboost, short for adaptive boosting, is a machine learning meta-algorithm introduced by Freund et al. [13] specialized for training supervised binary classifiers. This can be extended to multi-class problems using a binary decision tree introduced by Madzarov et al. [14]. Adaptive boosting minimizes training error through optimal feature selection for the dataset.

IV. Results

Our trajectory dataset consists of each agent's personal geolocation features extracted from the simulated GPS coordinates along with the agent's social context (friend and neighbor geolocation information). Using our agent-based simulation system, we created schedules and paths for 2000 agents over 28 days which resulted in a total of 11,406,956 points. 8,555,217 data points (21 days of observations) were used for training and the rest for testing the performance of the destination prediction.

We evaluated the performance of the destination prediction using gradually increasing partial trajectories, ranging from 10% to 90%. Our primary focus is to improve the accuracy of predicting from the early partial trajectories (10% to 30%).

Our experimental conditions included the following:

- User Trajectory: using personal geolocation features only;
- UT+Neighbor: personal geolocation features plus neighbor features;
- UT+Friendship: personal geolocation features plus friend features;
- Neighbor+Friendship: only social context features (neighbor+friend);
- Proposed: all features (personal, neighbor, and friend).

The proposed system achieves a prediction rate of 85.6% on the training set (Figure 3). The use of the social context features results in an increase in the prediction of the early partial trajectories ($< 50\%$). In absence of personal geolocation data, the social context can be used to predict the agent's movement on late trajectories (Figure 4). Hence, even in the case that the device's GPS is malfunctioning it is possible to do some destination prediction with social context alone.

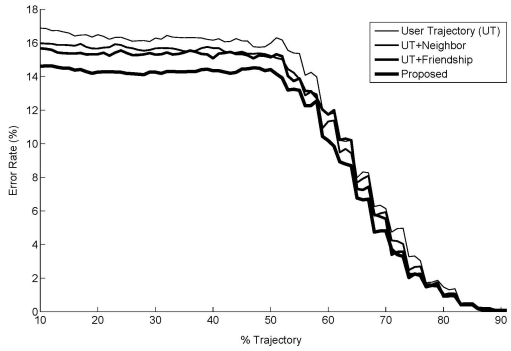


Figure 3. Average error rates for the destination prediction task over 80 possible destinations.

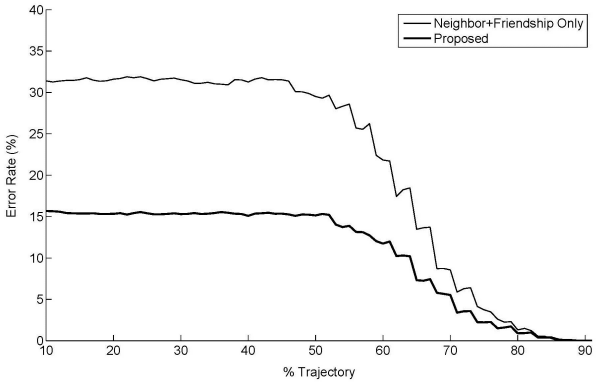


Figure 4. Comparison of the error rate of proposed method vs. the use of social context alone.

V. Conclusion

In this paper, we present a technique for learning a model of social context, based on aggregate information from two types of associates, *neighbors*, who are co-located spatially, and *friends*, who share a social connection. To evaluate the model, we created a social simulation of the schedule and travel patterns of 2000 people traveling between 80 destinations. We will release our social simulation to be used as a testbed for researchers studying the effects of urban design on human transportation behavior. A key advantage of our simulation is that it facilitates the modeling and study of human behavior patterns over a large urban area without the issues of biased sampling that can result from standard recruitment and data collection techniques.

Using Adaboost, we demonstrate that our model makes destination predictions with 85% accuracy from only 10% of the total trajectory by incorporating the proposed measures of aggregate social context. Even without any trajectory information, the aggregate social context alone can be used to predict the user's final destination at later stages in the journey. Although our implementation uses supervised classifiers, our social context features can be easily added to other types of prediction models that use probability maps or state estimation to track human movements. Although our technique is not applicable to standalone applications or devices, it indicates that the data sharing that occurs within popular social media applications can be leveraged in a relatively straightforward manner to improve individual user models.

References

[1] T. Schelling, "Dynamic models of segregation," *Journal of Mathematical Sociology*, vol. 1, pp. 143–186, 1971.

[2] J. Reif and H. Wang, "Social potential fields: a distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, no. 3, 1999.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of AAMAS*, 2001.

[4] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proceedings of AAMAS*, 2002.

[5] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," in *Proceedings of Ubicomp*, 2006.

[6] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines," in *Proceedings of AAAI*, 2004.

[7] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (TRIP): Incorporating personalization into route planning," in *Proceedings of IAAI*, 2006.

[8] L. Liao, D. Fox, and H. Kautz, "Extracting places and activities from GPS traces using hierarchical conditional random fields," *International Journal of Robotics Research*, vol. 26, pp. 119–134, 2006.

[9] B. Ziebart, A. Maas, J. A. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of AAAI*, vol. 3, 2008, pp. 1433–1438.

[10] N. Eagle and A. Pentland, "Eigenbehaviors: Identifying structure in routine," in *Proceedings of Ubicomp*, 2006.

[11] N. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. Campbell, and F. Zhao, "Enabling large-scale human activity inference on smartphones using community similarity networks," in *Proceedings of Ubicomp*, 2011.

[12] Mapquest, "Mapquest open directions API web service," March 2012, http://developer.mapquest.com/web/products/open/directions-service.

[13] Y. Freund and R. Schapire, "A short introduction to boosting," *Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[14] G. Madzarov, D. Gjorgjevikj, and I. Chorbev, "A multi-class SVM classifier utilizing binary decision tree," *Informatica*, vol. 33, pp. 233–241, 2009.

Online Learning of User-Specific Destination Prediction Models

Erfan Davami
Department of EECS
University of Central Florida
Orlando, Florida, 32816
Email: erfand@knights.ucf.edu

Gita Sukthankar
Department of EECS
University of Central Florida
Orlando, Florida, 32816
Email: gitars@eecs.ucf.edu

***Abstract*—In this paper, we introduce and evaluate two different mechanisms for efficient online updating of user-specific destination prediction models. Although users can experience long periods of regular behavior during which it is possible to leverage the visitation time to learn a static user-specific model of transportation patterns, many users exhibit a substantial amount of variability in their travel patterns, either because their habits slowly change over time or they oscillate between several different routines. Our methods combat this problem by doing an online modification of the contribution of past data to account for this drift in user behavior. By learning model updates, our proposed mechanisms, *Discount Factor* updating and *Dynamic Conditional Probability Table* assignment, can improve on the prediction accuracy of the best non updating methods on two challenging location-based social networking datasets while remaining robust to the effects of missing check-in data.**

I. Introduction

Mechanisms for learning predictive models of human transportation patterns are often foiled by the conflict between two forces: 1) strong correlations between destination and visitation time 2) long periods of disruptions when regular habits are not observed. People are often at work at 10:00 am, in bed at 1:00 am, and have numerous regular periodic commitments. This characteristic can dominate the error metric on the training set, and most feature selection paradigms will identify time and day as important features for predicting destinations.

However, there always exist long periods of disruption when regular habits are not observed. Users go on trips, experience deviations in their work and home routines, or change their lifestyles. In some cases, their behavior patterns will return to the learned baseline, but often the disruption represents a permanent change. During this period of time, visitation time and temporal dependencies will not be informative, and overreliance on those cues is punished. In this case unless the model can adapt to these changes in behavior, the accuracy will plummet since the majority of samples will be predicted incorrectly. In the case of a non-adaptive model, the learning mechanism will attempt to learn the model that predicts the majority of the samples, effectively sacrificing the samples that occur during those period of time.

To combat this problem, we introduce methods for online learning of user-specific destination prediction models, *Discount Factor* updating and *Dynamic Conditional Probability Table* assignment. The key to our methods is the use of efficient online updating procedures that modify the contribution of past data to the current prediction of the user's behavior. The baseline non-adaptive learning mechanism used in this paper is a Bayes net, which for these location-based social networking datasets achieves comparable performance to the a set of specialized methods for modeling human mobility [1]. Our two adaptation mechanisms perform online modifications of the conditional probability tables used for the inference to model the user's current transportation patterns; however the ideas behind the adaptive mechanisms could be generalized to other types of classifiers as well. This paper demonstrates that the use of online adaptation can offer significant improvements in prediction accuracy, particularly for users with certain mobility profiles.

This paper is organized as follows. Section II presents a selection of related work on learning models of human transportation patterns. Section III describes the location-based social media datasets and our proposed online learning methods. In Section IV, we present an evaluation of our proposed methods against several specialized methods for learning human mobility patterns before concluding the paper.

II. Related Work

Learning techniques that leverage temporal dependencies between subsequent locations can perform well at modeling human transportation patterns from GPS data. Although the assignment of GPS readings to road segments can be a noisy process, GPS generally provides a good continuous stream of data that can be used to learn a variety of models such as dynamic Bayesian networks [2], hidden Markov models [3], or conditional random fields [4]. The problem can also be formulated as an inverse reinforcement learning problem [5] in which the users are attempting to select trajectories that maximize an unknown reward function. Another predictive assumption that can be made is that the users are operating according to a steering model that minimizes velocity changes; this model can be combined with hidden state estimation techniques to predict future user positions [6].

However, in this paper, the datasets that we are using contain user check-ins collected from defunct location-based social networking sites (part of the Stanford Large Network Dataset

Collection [7]). Unlike in the Reality Mining dataset [8] or the Microsoft Multiperson Location Survey (MSMLS) [9], the user must voluntarily check-in to the social media site to announce his/her presence to other users. If the user doesn't check in, no data is collected. Thus, there are often significant discontinuities in the data when the user neglects to check in, and it is likely that the users opt to underreport their presence at certain locations. For this type of dataset, we found that the dynamic Bayes network which utilizes temporal dependencies actually performs slightly worse than the simple Bayes net used as the baseline for our model.

Rather than trying to learn temporal dependencies, our aim is to use the visitation time as the key feature, which is less sensitive to discontinuous data but very sensitive to local changes in the users' habits. These patterns can be discovered by doing an eigendecomposition analysis of the data [10], and interestingly can be predictive of users' activities several years into the future as shown in [11]. Cho et al. [1] demonstrate that a large section of this dataset can be fitted using a two-state mixture of Gaussians with a time-dependent state prior (Periodic Mobility Model), which we use as one of our comparison benchmarks; the two latent states in their model correspond to the user's home and work locations. The main contribution of this paper is to demonstrate how online learning can improve destination prediction by making the learned models more robust to temporary disruptions in user behavior patterns.

III. Method

This section describes:

1) the location-based social network datasets used to learn and evaluate our destination prediction models;
2) our baseline non-adaptive Bayes net model;
3) our first proposed method, *Dynamic Conditional Probability Table* assignment (DCPTA), for creating multiple region-specific models for each user;
4) *Discount Factor* adaptation (DF), our second proposed method for diminishing the effects of stale data in the conditional probability tables with a discount factor.

A. Datasets

The datasets used in this research were extracted from two location-based social networking websites called Gowalla and Brightkite. Cho et al. [1] have made both datasets publicly available at the Stanford Large Network Dataset Collection [7]. Gowalla (2007-2012), gave the users the option to check in at locations through either their mobile app or their website, and Brightkite was a similar social networking website that was active from 2007 to 2011. The data from these two websites consists of one user record per check-in that stores the user ID, exact time and date of the check-in, along with the ID and coordinates of the check-in location. Table I shows some features of these datasets, and Figure 1 shows a map of user activity within the United States.

Fig. 1. The scope of user check-ins across the United States for the Brightkite location-based social networking dataset. This location-based service was primarily active in the United States, Europe, and Japan between 2007 and 2011.

TABLE I
Location-based Social Media Datasets

Dataset	Gowalla	Brightkite
Records	6,442,857	4,492,538
Users	107,092	50,687
Average check-ins per user	60.16	88.63
Median check-ins per user	25	11

B. Baseline Model

For our non-adaptive model, we implemented a simple Bayes net with our modified version of the Bayes Net toolbox in Matlab. A Bayes net is a probabilistic graphical model that represents random variables and their conditional dependencies in the form of a directed acyclic graph. Figure 2 shows the Bayes net structure that we identified after experimenting with other more complicated model structures and dynamic Bayes networks in which the variables were conditioned on their values from the previous time step.

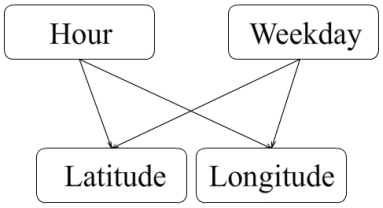


Fig. 2. Structure of the Bayes net used as the baseline model for inferring the user's latitude and longitude from the check-in day and time.

In this paper we use a fast simple method for training the network and extracting the most probable values of the output variables (the latitude and longitude nodes). The data structure of the network consists of $h \times d \times l$ matrices for the CPTs (Conditional Probability Tables) in which h and d are respectively the hour and day of the week at which the observation occurs and l is the list of possible check-in locations. For parameter learning, the corresponding cells of the CPT of the output nodes are incremented; predictions

are made by looking up the argmax latitude and longitude values for the user's location based on the check-in time. This method is feasible given the simple independence assumptions in this model and the large size of the dataset.

The main problem with the non-adaptive model is the large distortions which occur in the probability table when the user makes a long-range trip. Imagine a particular user being at some specific location, and following a repetitive pattern of activities for some months. If the user goes on vacation for a month, then the non-adaptive model will deliver a series of incorrect predictions based on the previously learned CPT, only slowly adapting to the new situation. Even once the user is back from the vacation, the effect of the probability distortion (caused by check-ins during the trip) is still clearly visible. We propose two new online learning algorithms capable of overcoming this problem, described in the next sections.

C. Dynamic Conditional Probability Table Assignment (DCPTA)

The movement pattern of most users in the dataset consists of a regular pattern of periodic short-range movements punctuated by occasional long-range movements. Figure 3 shows the movement pattern of one randomly selected user in the Brightkite dataset.

The average distance between subsequent check-ins ends up being a good measure of the user's mobility. When the user's movement exceeds twice the average distance between check-ins, it generally signals the start of a new mobility pattern. DCPTA (Dynamic Conditional Probability Table Assignment) uses this measure to determine when to learn a new user profile. By dividing the data into sections each time this jump in movement occurs, we can segment the movement of any user into sections with a relatively low variance which are stored in separate conditional probability tables and can be recovered if the user returns to those regions. Algorithm 1 describes how the DCPTA algorithm works.

Data: Check-ins of a particular user
Result: Dynamic Conditional Probability Table Assignment
Let $\mathcal{D}$ be the set of observed check-in distances
Let $\mathcal{S}$ be the set of observed stored segments
for *every new check-in* **do**
  Determine the distance of the current check-in from the initial check-in;
  $d_i = \text{dist}(coordinates_i - coordinates_0)$;
  if $d_i \leq 2 * mean(\mathcal{D})$ **then**
    load $CPT(\arg\min_{s \in \mathcal{S}} |d_i - s|)$
  else
    $\mathcal{S} \leftarrow \mathcal{S} \cup CPT(d_i)$;
  end
end

Algorithm 1: DCPTA (Dynamic Conditional Probability Table Assignment). This algorithm maintains a running average of the user's movements relative to an initial location and creates a new location-specific conditional probability table whenever the user's relative movements exceed a certain threshold.

D. Discount Factor Adaptation (DF)

DCPTA is most effective when the user returns to regions governed by previously learned conditional probability tables, and least effective when the user keeps changing his/her habits. For instance, users who are unemployed have a greater flexibility in their daily schedule which translates into a data series with a less defined temporal structure. To learn prediction models for users that exhibit erratic check-in behaviors, we introduce a discount factor, γ, into the process of updating the CPT such that the existing entry is discounted before incrementing the entry for the new observation. γ can range between 0 and 1; our results indicate that the use of the discount factor improves the online learning but that the learning is relatively insensitive to the magnitude of the parameter. Algorithm 2 gives the procedure for discounting conditional probability tables.

The discount factor reduces the effect of previous observations on the network, making the most recent check-ins more influential on the location prediction procedure. The advantage of this method compared to the previous proposed method is its lower computational and programming complexity. Applying the discount factor limits the location prediction to a few previous observations while discarding the stale data from older check-ins.

Data: Check-ins of a particular user
Result: Discounted Conditional Probability Table
for $\forall h, d, l$ **do**
  $CPT_{\text{Latitude}}(h, d, l) = *\gamma$;
  $CPT_{\text{Longitude}}(h, d, l) = *\gamma$;
end

Algorithm 2: DF (Discount Factor Adaptation). Before the CPT is updated with the incoming observation, the discounting procedure is applied. Discounting the conditional probability table reduces the effect of older check-ins on future predictions. This technique works well if the user's behavior changes slowly over time, rather than rapidly switching between destination-specific transportation patterns.

IV. Results

We employ two datasets, Gowalla and Brightkite, containing data from real users' check-in information [1]. Our evaluations are performed over the subset of users with greater than 100 check-ins, corresponding to 7600 and 8800 from Brightkite and Gowalla, respectively. We directly compare our methods against the techniques proposed by Cho et al. [1] and Gonzalez et al. [12].

As an additional baseline, we performed location prediction using the Bayes net (BN) described in Section 3. This network consists of the four nodes shown in Figure 2, where the predicted latitudes and longitudes are conditionally dependent

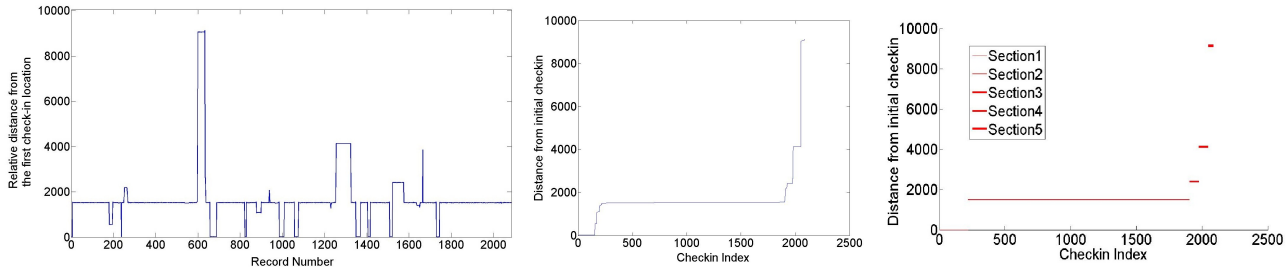


Fig. 3. Movement history of a user in the Brightkite dataset. (left) Initially, the latitude and longitude of the user's check-ins are converted to a single distance measurement relative to the first recorded check-in. (middle) The movement history can be divided into sections of low variance for learning the user's transportation pattern in a particular region by segmenting the data stream based on movement jumps that exceed twice the average distance between check-ins. (right) DCPTA learns a separate conditional probability table for each segment; these tables correspond to a different aspect of the user's routine.

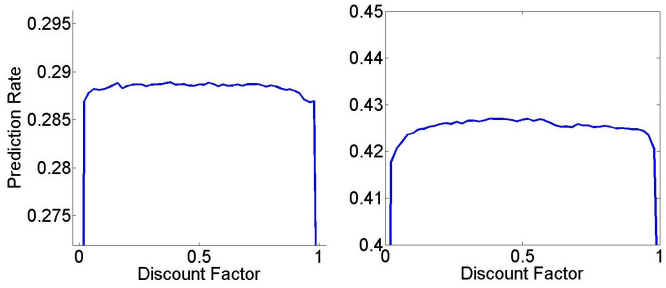


Fig. 4. Prediction performance using the proposed Bayes net vs. discount factor on the Brightkite dataset (left) and the Gowalla dataset (right). We observe that the prediction accuracy is relatively insensitive to the choice of discount factor and that a factor near 0.5 maximizes performance on both datasets.

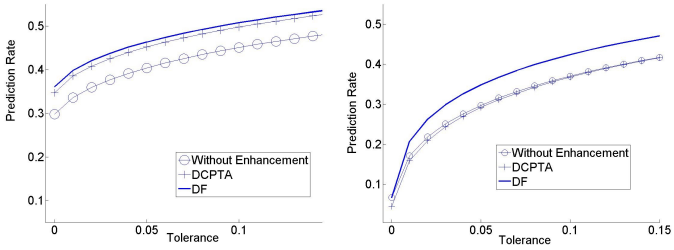


Fig. 5. Prediction performance of the Bayes net predictor and the proposed enhancements on the Bayes net (DCPTA and DF) on the Brightkite dataset (left) and the Gowalla dataset (right). Tolerance is the fraction of the total distance traveled by a user that is considered the acceptable distance of the prediction and the actual location of the user in every check-in.

on the weekday and hour of observation. For each user, the Bayes net is first trained using 15% of check-in data so that the Bayes net can gain some information about the periodic and geographical movement patterns of the user. The test results for this strategy and also the DCPTA strategy are shown in Figure 5.

A. Applying Discount Factor on the CPTs (the DF method)

As discussed above, applying a time-dependent discount factor can be a useful way of eliminating travel distortion over the conditional probability tables of our Bayes net. A discount factor of 0 implies a stateless system where counts are reset after each observation; conversely, a factor of 1 applies no temporal decay to the system. The first question we address is whether this discount factor dataset-specific, and how it impacts prediction accuracy. Figure 4 shows the effect of varying the discount factor from 0 to 1. We observe that the performance is relatively insensitive to the precise value of the discount factor and that a discount factor of 0.5 maximizes prediction accuracy for either dataset and is used in subsequent experiments.

Figure 5 shows how the prediction rate of the original Bayes net improves with the enhancement of dynamic conditional probability table assignment (DCPTA) and discount factor (DF). Specifically, we examine how accuracy varies with tolerance, which is defined as the level of error that is acceptable (considered as correct), expressed as a fraction of the total distance traveled by the user. For instance, a tolerance of 0.05 specifies that a prediction must lie within 5% of a check-in to be counted as correct. In this figure, we see that the proposed enhancements improve over the baseline BN over the entire curve, and add approximately 5% to the prediction rate, with DF slightly outperforming DCPTA, over the entire curve.

Figure 6 compares the location prediction results from our methods to the following five recent methods described in the literature:

1) Periodic mobility model [1], denoted as PMM;
2) Periodic and social mobility model [1], denoted as PSSM;
3) Gaussian Mixture Model [12], denoted as G;
4) Last-known location model [1], denoted as RW;
5) Most frequent location model [1], denoted as MF.

The *Periodic Mobility Model (PMM)* assumes the majority of the human movement in a network is based on a periodic movement between a small set of locations. The *Periodic and Social Mobility Model (PSMM)* also adds additional parameters to model movement driven by one's social relationships with other members of the network.

Our Bayes net methods are denoted as BN, BN&DCPTA, and BN&DF, and the comparison employs a tolerance level of 2.7%.

We observe that the BN without enhancement (36%) per-

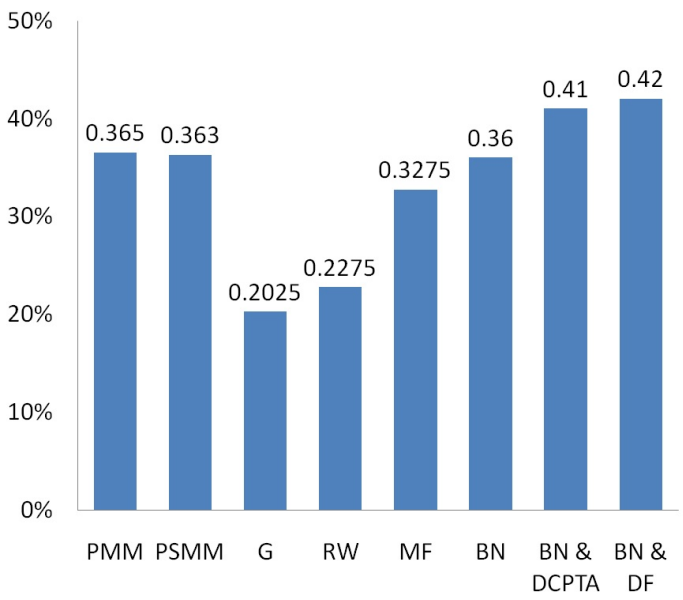


Fig. 6. Prediction performance of the Bayes net predictor and the proposed enhancements on the Bayes net along with the performance of prior art on the Brightkite dataset.

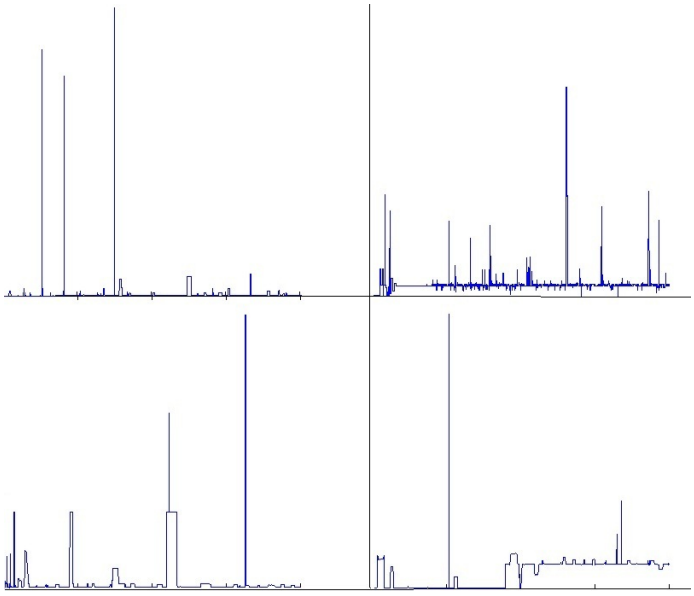


Fig. 7. Comparing the movement pattern of different users in the Brightkite dataset. The top two patterns are better predicted using the DCPTA method however the DF method performs better at predicting the bottom two patterns. A possible hypothesis is that DCPTA performs better for users who have multiple short trips, compared to the DF updating.

forms almost as well as the best of the state-of-the-art approaches, PMM (36.5%) and PSMM (36.3%). However, with our enhancements, we see that accuracy increases by almost 6%, with BN&DF at 42%, slightly outperforming BN&DCPTA at 41%. The remaining baselines (B, RW, MF) are not competitive.

The results shown in Figure 6 are averages over many predictions. Figure 7 provides a more detailed look at some specific instances, and we see that DCPTA does occasionally outperform DF on users with certain features. The top row of the figure shows examples of users for whom DCPTA performs best while the bottom row shows some for whom DF is a better predictor. We hypothesize that users who spend time shuttling between a small set of locations and relatively little time on infrequent long-range trips are better predicted using DCPTA; conversely, DF is better able to handle users who go on long trips and make frequent check-ins away from home.

B. Handling Missing Data

Missing data within the dataset can be a severe problem for location prediction algorithms. The algorithms used for prediction of GPS data often will not work as well when dealing with check-in data due to the high inconsistency of datapoints. Unfortunately, due to the relatively high overhead imposed on users by a check-in action, the chance of collecting data with missing check-ins is inevitable.

In this section we examine the robustness of the proposed algorithms towards missing data. Seven experiments were conducted using both datasets in which a percentage of check-in data was randomly withheld from the dataset. Figure 8 summarizes the prediction results on each dataset. All of the

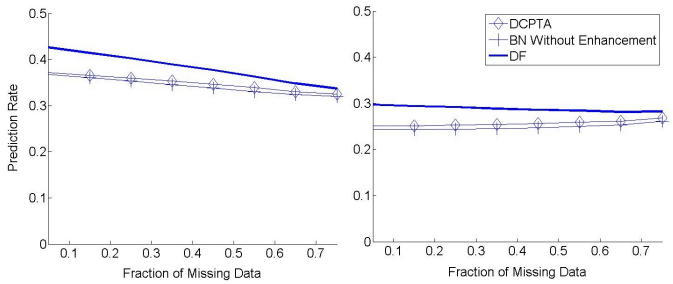


Fig. 8. Prediction rates of proposed algorithms when applied to datasets with missing data. Some fraction of the check-in data is randomly withheld and then predicted using the belief network and the proposed enhancements. Our approaches exhibit robustness to missing data.

proposed methods are quite robust to missing data, with the best (DF) showing a drop of only 10% for 70% missing data on the Brightkite dataset (left) and negligible loss on the Gowalla dataset (right). This confirms our belief that there is significant redundancy in the second dataset that can be exploited. Somewhat surprisingly, we observe a slight *improvement* in DCPTA's performance with missing data on the Gowalla dataset. We attribute this to the fact that withholding data has the effect of reducing check-ins corresponding to long-range travels, which results in a reduction of such outliers (Fig. 7).

C. Complexity

We briefly summarize the computational complexity and storage requirements for the proposed methods. The core data structure behind our methods is a conditional probability table (described in Section 3B). Storing such a discretized table,

TABLE II
COMPUTATIONAL TIME REQUIRED FOR EACH DATASET (MINUTES)

Name of Dataset	Gowalla	Brightkite
Processed Users	8800	7600
Processed check-ins	2,694,344	3,399,651
Belief Network	9	12
BN&DF	10	12
BN&DCPTA	19	20

even at double-precision, is cheap: a table with $24 \times 7 \times 700$ double-precision cells requires less than 1MB of memory.

Conditional probability tables are also computationally efficient, affording constant-time updates. Finding the maximum in the table employs an exhaustive scan that is linear $O(N)$ with respect to the number of cells, N; in practice, since the number of cells is around 100K, this remains very efficient.

The DCPTA method (Section 3.C) requires multiple CPTs for every segment of the users' movement pattern, thus requiring a memory growth of $O(s)$ where s denotes the number of segments. In terms of computational complexity, the algorithm must search s CPTs in order to load the right CPT for future use, resulting a computational complexity of $O(sN)$.

Finally, the DF method simply multiplies the CPT by a real number (discount factor). This procedure has no impact on the memory usage of the belief network however, but increases the computational complexity equivalent to a scalar matrix multiply, which is theoretically $O(N)$ but very efficient on current hardware.

Table 2 presents measured running times for each method on both datasets.

The processing was done using Matlab 2012a, an Intel Quad-core Xeon Processor and 18GB of memory.

V. CONCLUSION

In this paper we present two new algorithms for online learning of user-specific destination prediction models, Dynamic Conditional Probability Table Assignment (DCPTA) and Discount Factor updating (DF). Although we describe the use of our online update procedures for a Bayes net model, the same intuitions behind the discounting of stale data and threshold switching between multiple models can be applied toward online learning procedures for other types of classifiers. Our proposed destination prediction model leverages the predictive power of visitation times while rapidly adapting to schedule changes by the users. Adapting to changing user habits allows our model to achieve better predictive performance than the best static models which are continually penalized by non-stationary user behavior.

ACKNOWLEDGMENT

This research was funded by AFOSR YIP FA9550-09-1-0525 and DARPA award N10AP20027.

REFERENCES

[1] E. Cho, S. Meyers, and J. Leskovec, "Friendship and mobility: user-movement in location based social networks," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2011.

[2] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines," in *Proceedings of National Conference on Artificial Intelligence*, 2004.

[3] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (TRIP): Incorporating personalization into route planning," in *Proceedings of Conference on Innovative Applications of Artificial Intelligence*, 2006.

[4] L. Liao, D. Fox, and H. Kautz, "Extracting places and activities from GPS traces using hierarchical conditional random fields," *International Journal of Robotics Research*, vol. 26, pp. 119–134, 2006.

[5] B. Ziebart, A. Maas, J. A. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of National Conference on Artificial Intelligence*, vol. 3, 2008, pp. 1433–1438.

[6] B. Tastan and G. Sukthankar, "Leveraging human behavior models to improve path prediction and tracking in indoor environments," *Pervasive and Mobile Computing*, vol. 7, pp. 319–330, 2011.

[7] "Stanford Large Network Dataset Collection," http://snap.stanford.edu/data/index.html.

[8] N. Eagle and A. Pentland, "Reality mining: Sensing complex social systems," *Journal of Personal and Ubiquitous Computing*, 2005.

[9] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," in *Proceedings of the ACM International Conference on Ubiquitous Computing*, 2006.

[10] N. Eagle and A. Pentland, "Eigenbehaviors: Identifying structure in routine," in *Proceedings of the ACM International Conference on Ubiquitous Computing*, 2006.

[11] A. Sadilek and J. Krumm, "Far out: Predicting long-term human mobility," in *Proceedings of National Conference on Artificial Intelligence*, 2012.

[12] M. Gonzalez, C. Hidalgo, and A. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

Extracting Agent-Based Models of Human Transportation Patterns

Rahmatollah Beheshti
Department of EECS
University of Central Florida
Orlando, Florida 32816
Email: beheshti@knights.ucf.edu

Gita Sukthankar
Department of EECS
University of Central Florida
Orlando, Florida 32816
Email: gitars@eecs.ucf.edu

***Abstract*—Due to their cheap development costs and ease of deployment, surveys and questionnaires are useful tools for gathering information about the activity patterns of a large group and can serve as a valuable supplement to tracking studies done with mobile devices. However in raw form, general survey data is not necessarily useful for answering predictive questions about the behavior of a large social system. In this paper, we describe a method for generating agent activity profiles from survey data for an agent-based model (ABM) of transportation patterns of 47,000 students on a university campus. We compare the performance of our agent-based model against a Markov Chain Monte Carlo (MCMC) simulation based directly on the distributions fitted from the survey data. A comparison of our simulation results against an independently collected dataset reveals that our ABM can be used to accurately forecast parking behavior over the semester and is significantly more accurate than the MCMC estimator.**

I. Introduction

Agent-based simulations have been used successfully for modeling human social systems in diverse fields including economics, sociology, anthropology, and archaeology [1]. A perennial question that arises in the development of an agent-based simulation is how to initialize the models to create a realistic population of agents. In simple models with few parameters, it is feasible to perform a sensitivity analysis to explore the effects of the parameters on the performance of the simulation. However in more complicated agent decision-making models, creating a realistic population of agents can be challenging due to the larger range of parameters governing the behavior of the simulated entities.

Surveys and questionnaires can be used to collect an accurate static snapshot of the behavior of large social systems but lack the predictive power of simulations. It is more difficult to explore "what-if" questions with a survey since posing questions to participants about hypothetical scenarios can be problematic due to human cognitive biases such as anchoring or risk-aversion. In this paper, we show how both methodologies, surveying and agent-based simulation, can be combined to model human social systems with higher verisimilitude and to explore the ramifications of different behavior patterns and trends.

This paper specifically addresses the problem of creating individual agent profiles for an activity-based microsimulation model of transportation, dining, parking, and building occupation preferences on a large university campus. One problem with agent-based models is that linking the models and simulation processes with the observed data is challenging. The main contribution of our research is to demonstrate a procedure for systematically linking the observed survey data of people's transportation preferences with an executable agent model. In contrast, stochastic simulation approaches such as Markov Chain Monte Carlo (MCMC), have been used to forecast the outcome of temporal processes and are simple to create and initialize from observed data [2]. However, in our results, we show that our method is substantially more accurate at forecasting future effects than an MCMC estimator initialized from the same survey data, even at answering relatively simple questions. An additional benefit is that manipulating the operation of an agent-based model can empower researchers with better intuitions about the reasons behind emerging group phenomena rather than merely observing the unfolding of a stochastic process [3].

Urban simulation is a particularly fertile area for agent-based simulation research since it requires modeling a large number of interdependent agents making sequential decisions within a small region. Benenson et al. [4] present two motivations for defining *urban agents*, as a distinct group within the general class of autonomous agents:

1) urban agents often have a high degree of mobility resulting in rapidly changing spatial relationships.
2) to succeed, urban agents require a strong capability to perceive and adapt to the evolving urban environment shaped by neighboring agents.

In a general urban model, there can be many classes of agents—developer agents constructing new buildings, car agents moving in traffic, business agents providing services to customer agents, and land-use agents who own and manage parcels and lots [4]. In our model, we focus on modeling transient activity patterns such as transportation habits, dining preferences, and building occupation times. The goal is to predict the large-scale aggregate activity patterns of thousands of students over the duration of the semester, in contrast to work that has been done on learning individual transportation modality and route preferences using cell phone and GPS data from hundreds of individuals (e.g., the MIT Reality Mining

project [5] or the Microsoft Multiperson Location survey [6]). An alternate approach, crowdsourcing, leverages the "wisdom of the crowd" to answer simple queries and has been demonstrated to be a useful tool for gathering specialized real-time data for various transportation related activities such as gas pricing (GasBuddy) or parking spot detection (OpenSpot). It can be a useful replacement for questionnaires and surveys in cases where some incentive exists for users to install software and self-report on their behavior. These technologies are highly complementary, and in this paper we demonstrate how the models from our activity-based microsimulation can supplement mobile device monitoring and crowdsourcing, enabling accurate transportation forecasting and exploration of hypothetical scenarios.

This paper is organized as follows. In the next section, we describe the process of extracting agent-based models from survey data and our activity-based microsimulation. Then we describe the construction of a benchmark Markov Chain Monte Carlo simulation in Section III. Section IV presents an evaluation of our proposed agent-based simulation initialization model on a simple parking forecasting problem. We conclude by describing other related work in simulating urban social systems and transportation forecasting.

II. Method

In this section, we describe the development process for our activity-based microsimulation, including the agent-based model, survey data collection, activity profile generation, path planning, and simulation system. For our urban region, we selected the University of Central Florida main campus, which is one the biggest academic institutions in the US with almost 59,000 students and 10,567 staff. It is adjacent to the Central Florida Research Park which is home to 116 companies with approximately 9,500 employees. The presence of nearby businesses and existence of commuters traveling between multiple UCF campuses give rise to a social system with a diverse and complex set of transportation patterns.

A. Data Collection

To simplify the data collection process, our initial study focused solely on modeling student transportation, dining, and building occupancy patterns. 1003 students responded to our online survey posted on `KwikSurveys` which was advertised on various campus email lists. The questions on the survey were grouped into six different categories, related to possible places that could be visited on the main campus:

1) Daily attendance patterns, including the days and times that the participant arrives and departs the main campus
2) Initial location, either the dorm (for on-campus students) or the entrance that was used to enter the campus (for commuting students)
3) Visitation frequency for on-campus dining locations
4) Usage patterns for recreation and athletic facilities
5) Usage of administrative and other miscellaneous locations
6) Frequency of parking lot and shuttle stop usage

For categories three through six, students were specifically queried about their visitation frequencies. For these questions, responses included one of: *never*, *rarely*, *once a month*, *several times in a month*, *once a week*, *several times in a week* and *every day*.

In addition to the survey data, our agent-based simulation used publicly available statistics about UCF* and the main campus building map†. A graph of the campus paths and roads was created from the main campus building map. The set of nodes in the graph is the union of the locations in the survey plus the junctions between the streets and pathways. The edges of this graph represent the roads and walkways among the nodes. The weights of the edges show the distance between the connecting nodes. Each node and edge has a tag. This tag for the nodes indicates whether they are a location of interest on the map or merely a junction. For example, a department is a location of interest, and a junction created by intersecting two roads or walkways is not. The tag for the edges determines whether they are walkways or roads. Figure 1 shows a snapshot of the map and also the corresponding graph in the background.

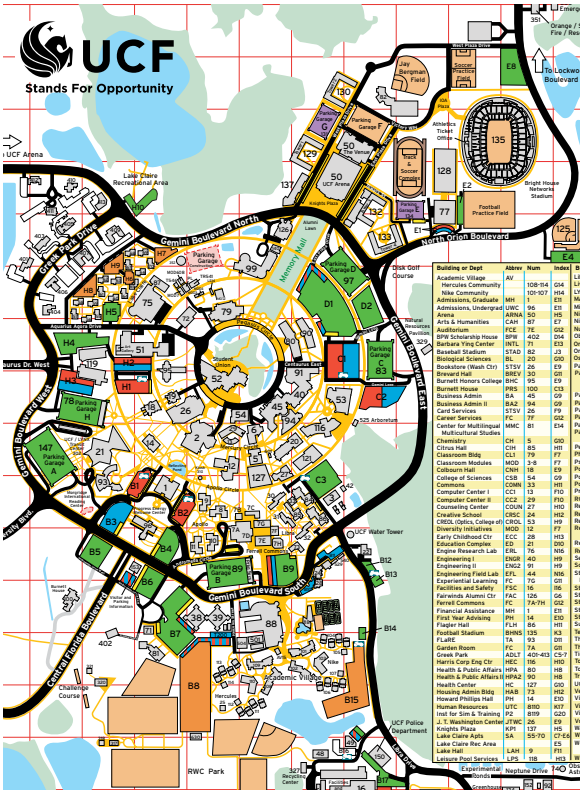
B. Agent-Based Model

To perform transportation forecasting on the UCF campus, we created an agent-based model for simulating the common activities (transportation, dining, recreation, and building occupancy) performed by the 47,000 students on the main campus. Each agent in the model represents an individual student and has a unique set of parameters that govern his/her activity profile. An agent's defining parameters are: *entrance*, *dormitory*, *department*, *class building*, *arrive*, *depart*, *lunch*, *dinner*, *beverage*, *recreation and wellness*, *parking*, *shuttle*, and *miscellaneous*. The first four parameters designate the single (most common) value of the agents' entry point to the campus, housing situation, home department, and main class building. Note that we did not explicitly represent the students' class schedules in the model. Even though this would have improved the fidelity of the model, we felt that addition would not generalize well to other types of urban models. *Arrive* and *depart* are lists showing the times the agent enters the campus and leaves it. The remaining parameters are lists of locations for the agent's dining, recreation, and commuting. Additionally, each parameter that includes a location has another matching parameter that shows the time or frequency of visiting that location.

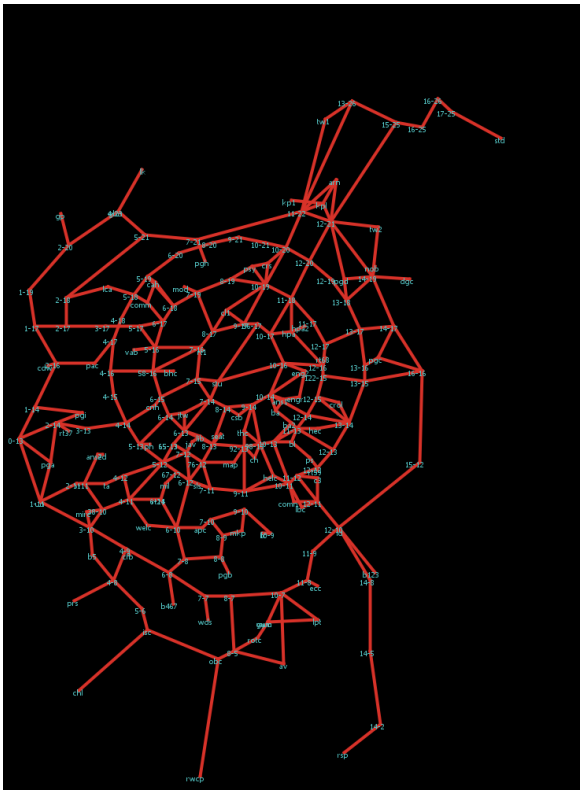
In this paper, we explore two agent-based modeling methods:

- **ABM only:** agent model parameters are randomly sampled from a uniform distribution over a realistic range of values. This method is commonly used in a most of the ABM systems described in our related work overview and is used as a benchmark for our proposed method.

*http://www.iroffice.ucf.edu/character/current.html
†http://map.ucf.edu/printable/



(a) UCF main campus



(b) Simulation graph

Fig. 1: The map used in the simulation along with the corresponding graph. Gray spots are buildings, black lines show the campus roads, and yellow lines indicate the walkways. Parking lots are marked in green (student), blue (staff), and red (faculty).

- **ABM+survey:** agent model parameters are randomly sampled from a set of continuous and discrete distributions that correspond to responses to survey questions. The parameters for these distributions are selected based on the best fit of the survey data.

Rather than directly mapping the survey data to simulated entities that match the exact preferences of one of the survey respondents, we attempt to learn a general model of the population by fitting a statistical distribution to the answers of every question. For those questions that were related to the time of visiting a location (e.g., campus arrival and departure times), a Gaussian distribution was used to create a continuous distribution of arrival and departure times for the population of agents. For those questions where the respondents provided frequencies (e.g., how often campus dining locations were visited), we evaluated the performance of several discrete distributions and selected to the Poisson distribution as offering the best fit for most of the questions. Figure 2 shows the fit of all of the 79 distributions used to initialize our ABM; the better fit distributions have negative log likelihoods falling closer to zero, shown in the figure as the shorter bars. Note that for our ABM we opted to use the same distribution model for all questions of a certain category regardless of the fit, rather than attempting to optimize the fit of the observed data by changing the form of the model. There was no discernible

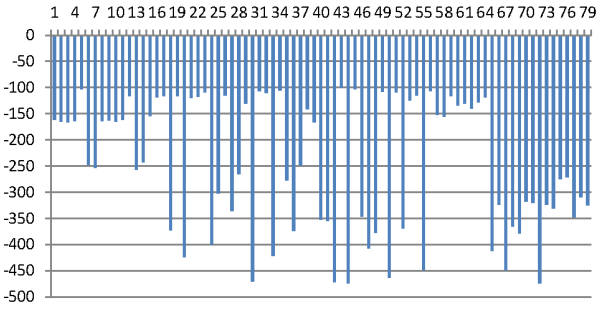
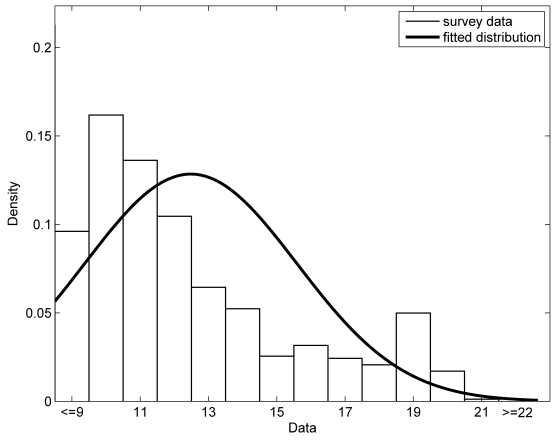


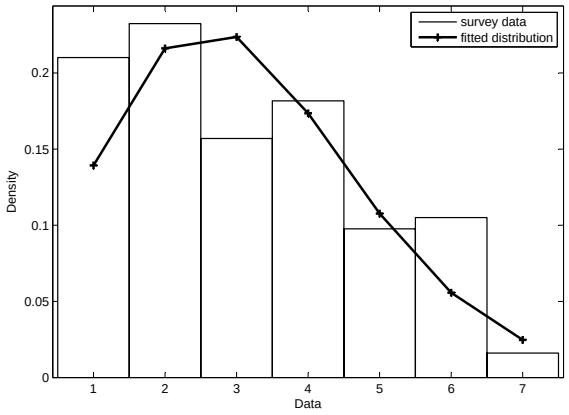
Fig. 2: The fit of the 79 distributions used to initialize the ABM (log likelihood vs. the question index). Better fit distributions have negative log likelihoods falling closer to zero, corresponding to a higher probability of the survey data being drawn from the distribution. The mean log likelihood over all the distributions is -244.1, with a standard deviation of 123.5.

interaction between the fit of the model by question category (e.g., dining patterns were better fitted by one model and parking lot preferences by another) so we used the same two distributions across all the questions.

After fitting the Poisson distribution on the qualitative data,



(a) Gaussian distribution for the entry time to the UCF main campus on Wednesdays. Columns correspond to the arrival time.



(b) Poisson distribution for the frequency of visiting dining locations in the Knights Plaza area of campus. Columns 1-7 are related to frequency of visits.

Fig. 3: Two fitted distributions used to initialize the agent populations in the proposed method (**ABM+survey**)

a mapping function is used to work with the values obtained. This function maps the qualitative frequencies to exact dates and times. Each term, from *rarely* to *everyday*, is treated separately. For instance, the term *rarely* is mapped to a random day in a 60 day period. Figure 3 shows two example distributions used to initialize the ABM for Wednesday campus entry times (Figure 3a) and the pattern of visiting dining locations in the Knights Plaza area of the campus (Figure 3b).

C. Activity-oriented Microsimulation

When the simulation commences, all the agents are initialized with parameters that remain constant over the lifetime of the agent and are used to create daily activity profiles. Our simulation is implemented in the *Netlogo* [7] environment. NetLogo (originally named StarLogo) is a high level platform, providing a simple yet powerful programming language, built-in graphical interfaces, and comprehensive documentation. It is particularly well suited for studying the evolution of complex systems over time [8].

In this environment, time is discrete and simulated by ticks where a tick is one unit of time. In our model, one tick represents one hour of activity in the real world. When the model starts, each agent runs within a loop. The loop continues until the simulation is stopped. Figure 4 shows the runtime process by which an agent activity profile is generated.

Based on the agent's parameters that are initialized at the beginning of the simulation, the agent activity profile generator determines what should an agent do and where should be at every time (tick). If sampling the agent's profile indicates that it should be on campus, then the function compares the current time with the possible activity times produced by the mapping function that maps frequencies from the agent's distribution model to specific times and dates. If a match is found, then the agent opts to travel to that location. Otherwise, the agent remains at its department as its default place. On the other hand, if the profile generator determines that the agent

```
switch current-time-status:
    case entrance-time
        if live-off-campus then
            enter-campus //go to one of the entrances
            go-to-parking-or-shuttle-stop
        end if
    case on-campus-time
        if should-go-somewhere then
            go-to-destination
        else
            stay-at-department
        end if
    case return-time
        if live-off-campus then
            go-to-parking-or-shuttle-stop
            leave-campus //go to one of the entrances
        else
            go-to-dorm
        end if
    case not-on-campus
        disable
```

Fig. 4: Runtime generation of agent activity profiles

shouldn't be on campus, then the agent goes to (or remains in) the disabled state.

Various constraints are checked before an agent decides to go to a place. These constraints ensure the consistency of the whole model with the real world facts. The main consistency checks are summarized below:

- **daily schedule**: whenever an agent's model generates a date and time for visiting a location on campus, it

checks the agent's arrival and departure times for that day. Campus activities that fall outside those boundaries are eliminated.

- **activity overlap**: whenever the agent's model generates trips that overlap in time, requiring the agent to be in multiple places at once, one of the overlapping tasks is shifted to a later time.
- **campus constraints**: known information about the operation hours of administrative offices, classroom buildings, and shuttle transportation is incorporated into the simulation. If the agent's model generates trips that violate the known operation hours, those trips are discarded.

A shortest path graph algorithm is used to choose the path that an agent should traverse between its start and end positions. To speed-up the model, an all pairs shortest path graph algorithm computes all of the shortest paths. A slightly modified version of Floyd-Warshall algorithm was used for this purpose. All path planning occurs at initialization; candidate paths are stored in a look-up table to be accessed later. The time complexity of Floyd-Warshall algorithm is $\theta(n^3)$. There are n^2 paths, and the length of each path is at most *n*, hence the space needed to store the paths (look-up table) is in the order of $O(n^3)$.

III. Markov Chain Monte Carlo Simulation

To compare the performance of our ABM model, we created a benchmark Markov Chain Monte Carlo simulation for making a limited set of forecasts based on the survey data. Markov Chain Monte Carlo describes a family of methods for performing Bayesian inferences using stochastic simulation. It has been used successfully in a wide variety of scientific [9] and engineering modeling applications [10]. MCMC allows us to draw samples from a distribution $\Pi(x)$ without having to know its normalization. With these samples, it is possible to compute any quantity of interest about the distribution of x, such as confidence regions, means, standard deviations, or covariance [11].

Rather than creating one large monolithic simulation of the entire urban system to explore a variety of scenarios, here MCMC is used to directly to forecast specific questions of interest, such as parking lot utilization. Our MCMC simulation uses the Metropolis-Hastings algorithm which randomly generates candidate points drawn from a proposal distribution around the existing points. It accepts candidate points with the following probability:

$$\alpha(x_e, x_c) = min(1, \frac{\Pi(x_c)q(x_e|x_c)}{\Pi(x_e)q(x_c|x_e)})$$

Here, x_e is an existing point and x_c is the candidate point. $\Pi(x)$ is the posterior distribution [‡] and $q(x)$ shows the proposal distribution. For more details about the MCMC method, the reader is referred to the following reference [2].

In this study, we have used the MCMC method as a benchmark to compare against the proposed agent-based model to demonstrate the benefits of combining the higher fidelity ABM with the survey data. MCMC is used to estimate the number of cars entering the parking lots at different times of a day. One can envision this as a two dimensional diagram with the horizontal axis corresponding to the time of a day, and the vertical one showing the number of cars entering a specific parking lot. The survey data from the questions about the attendance pattern and frequency of parking lot usage are used to initialize the MCMC model. Observations for the Bayesian inference process are simply obtained based on the results of the survey data for a simulation period of 90 days. Imagine that based on the survey data a student respondent enters the campus everyday before 9 am, leaves at 5 pm, and reports his general usage of parking lot A as being at a frequency of once a week. In this case, the expectation is that the student would have occupied Lot A twelve times $(90/7)$ during the simulation period so a corresponding number of samples tagged with the reported time range are produced and added to the input observation data.

The Metropolis-Hastings algorithm from the MCMC toolbox for Matlab [12] is used for the simulation. Our MCMC model assumes the prior is of the form of a Poisson distribution, the same as our **ABM+Survey** model. For the proposal distribution, a Gaussian is used. The MCMC attempts to find the most likely value of the the mean of the Poisson distribution (λ in $\frac{\lambda^x e^{-\lambda}}{x!}$).

[‡]More accurately stated, $\Pi(x)$ is a value proportional to the posterior distribution, the Bayes numerator.

Parameter	Value
Agents	47,000
Days	100
Time Range	07:00 - 24:00

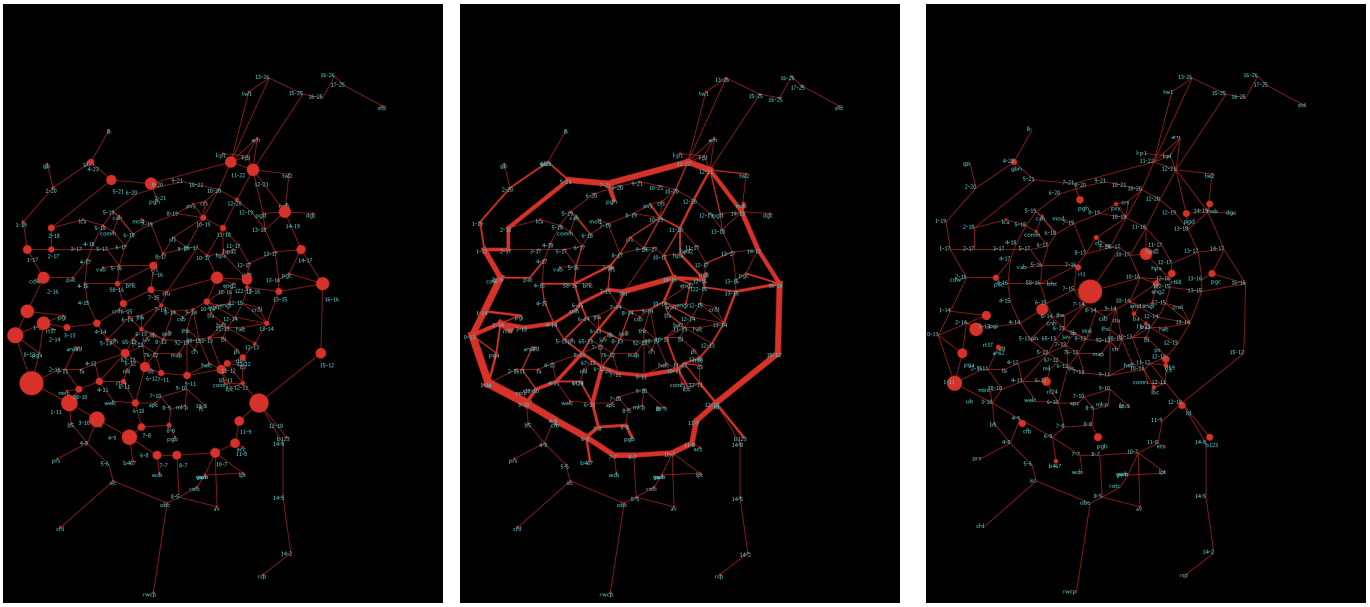
TABLE I: The parameter settings of experiments

IV. Results

To evaluate the performance of the agent-based model under different initialization conditions, we examined the transportation forecasts produced by the simulation, both through visualization and by comparing the predictions against a dataset collected by the UCF Parking Services office. The parameter values that are used in all of the experiments are listed in Table I.

One of the main applications of our microsimulation is analyzing pedestrian movement and car traffic on campus. Figure 5 shows the average visitation frequency for UCF campus locations (junctions, roads, and buildings) as predicted by our simulation. The size of the circles in 5a and 5c, and the thickness of line in 5b are proportional to the number of the agents who passed or visited these places.

Some obvious facts that can be easily verified by a domain expert are also observed in this set of results. For instance, as on most university campuses, the student union is the most frequently visited place since it is the venue for most events and many dining locations. The wide drivable boulevard that surrounds the campus dominates the road usage as it is the



(a) Junctions (b) Roads and walkways (c) Buildings

Fig. 5: Average traffic through different locations on the campus as predicted by the simulation. The simulation clearly shows several campus usage trends that are easily verified: 1) high usage of the circle road, the only drivable boulevard around the campus 2) high traffic at both main campus entrances (bottom left and right) 3) high student union usage (center), both of the building and incoming road 4) high traffic near the biggest parking lots on campus (two large circles in the bottom left).

only way that can be used by cars and shuttles to reach most points on campus.

A question of daily interest for most students is parking lot usage: which lots have vacancies and where can the best parking spots be found? UCF Parking Service performed a visual survey of lot usage in Fall 2011 and created a data set which we compared to our hourly microsimulation forecasts of student lot usage. Note that although we ask questions about parking preferences on the survey, the survey data alone is insufficient to directly reveal the hourly parking lot usage without the agent-based simulation or the MCMC.

Figures 6 and 7 show the microsimulation forecasts for the different student parking lots as predicted by: 1) **ABM Only**: the agent-based model initialized and simulated without survey data; 2) **ABM+Survey**: our proposed model in which the survey data is used to create the distributions for generating agent activity profiles; 3) **MCMC**: the Markov Chain Monte Carlo parking simulation 4) **Empirical**: the actual data provided by UCF Parking Services. The horizontal axis shows the names of the parking lots and the vertical the cars entering the lots during different time periods.

The results clearly show that the forecasts from the microsimulation are fairly close to the actual data collected for most of the lots. The one exception is the mismatch between the UCF Parking Service results and the forecast for Parking Lot A usage at 4pm. One likely explanation for the discrepancy is that an increase in student enrollment since the empirical data collection has caused a general increase in parking lot usage. The empirical data was collected in Fall 2011, while the survey and simulation data was gathered in the Spring 2012. Since Parking Lot A is a large, rarely fully occupied lot, it has a tendency to absorb overflow traffic. Another caveat is that the current version of our simulation does not model the movement patterns of the staff/faculty who are also allowed to park in student lots. An important

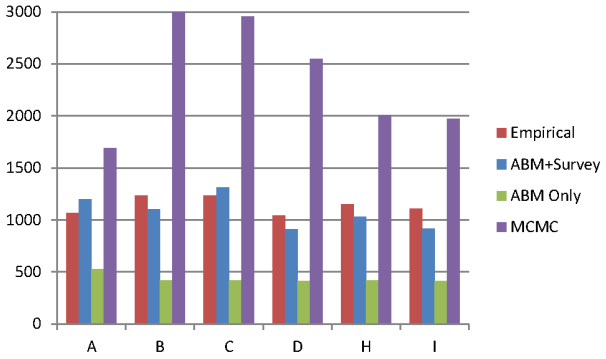


Fig. 6: The average number of cars entering the student parking lots at noon as predicted by the MCMC simulation (**MCMC**), the standard modeling method (**ABM Only**), our proposed method (**ABM+Survey**), and **Empirical**, the data from UCF Parking Services. Our proposed method is substantially more accurate at predicting parking lot utilization (as shown by the small difference between the red and blue bars) than the MCMC or the ABM without our initialization technique.

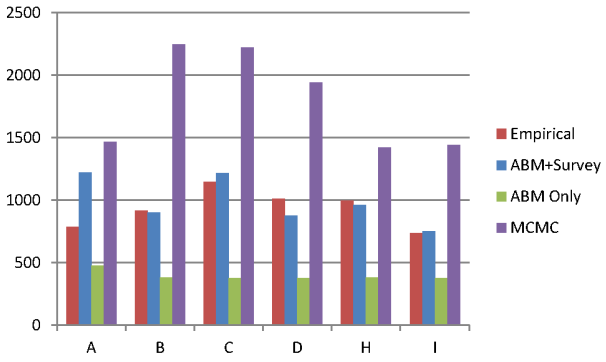


Fig. 7: The average number of cars entering the student parking lots at 4 PM as predicted by the MCMC based simulation (**MCMC**), the standard modeling method (**ABM Only**), our proposed method (**ABM+Survey**), and **Empirical**, the data from UCF Parking Services. Our proposed method is the most successful at predicting the usage of most of the parking lots, shown by the small difference between the red and blue bars. The ABM alone underestimates the parking lot usage, whereas the MCMC predicts the trend of usage patterns but overestimates the number of cars.

thing to note is that the ABM model alone, initialized with a reasonable set of initial parameters, does not do a good job at forecasting parking lot usage, which is a relatively simple question. Without information about the times that the students are likely to be found on campus or their lot preferences, it predicts a fairly even spread of cars to lots. Also since it does not accurately model the time peaks in campus usage, it tends to predict an even spread of lot usage across all day time hours. The MCMC, which is initialized with the same survey data as our proposed method but lacks the detailed activity-based microsimulation, consistently overestimates the parking lot usage. However, it is more successful at expressing general peaks and dips in the occupancy that are missed by the ABM alone. None of the models use specific parking lot physical constraints (such as maximum capacity) nor time constraints (banned parking times) that would give them an unfair advantage in their calculations.

V. Related Work

Agent-based models have been used to successfully model urban environments in a wide variety of applications including: 1) civil and environmental transportation analysis [13] 2) geographic information systems (GIS) for visualizing patterns and trends in spatial areas [14] and 3) archaeological studies of land site usage in ancient civilizations [15].

Although these systems do not necessarily have to accurately simulate physical interactions, incorporating spatial information and heterogeneity into agent-based models can improve our ability to draw conclusions about the behavior of complex systems in realistic environments, which may be different from conclusions drawn with artificial environments [16]. With the inclusion of GIS to represent a spatially, georeferenced environment, the impact of human behavior patterns can be linked to specific spatial locations and when used correctly can provide a powerful tool for policy makers and the public to understand the potential consequences of their decisions [17]. For instance, [18] presents an agent-based model for analyzing the influence of neighborhood design on daily trip patterns based on the detailed trip survey data from seven Traffic Analysis Zones (TAZs) in Ottawa, Canada. In [19], results obtained from a behavioral survey of driving behaviors were used to identify and fit a series of agent behavior parameters defining driver characteristics, knowledge and preferences; the authors also present a case study implementing a simple agent-based route choice decision model within a microscopic traffic simulation tool. However neither of those works present a systematic evaluation of different modeling techniques as was done here. In a different domain, modeling the diffusion of water-savings innovations, an ABM was calibrated using empirical data stemming from a questionnaire survey [20], showing that this technique can generalize across simulation domains.

In this paper, we describe the development of an activity-based microsimulation for modeling and forecasting transportation patterns on the UCF campus. For a complete survey of agent-based approaches to transportation and traffic management, the reader is referred to [21]. In many of these systems, each agent represents an individual person or vehicle, thus giving rise to the question of how to initialize the models to create behavior that is realistic in both the individual and aggregate sense. The four methods customarily employed are: 1) agents are randomly initialized using a reasonable range of parameters [13]; 2) recommendations from domain expert are used to guide parameter selection; 3) agents are designed to directly mimic actual members of the population [14]; 4) a hybrid combination of random initialization and expert guidance is employed at initialization [15]. In contrast, our simulation attempts to mirror the population using a series of fitted distributions rather than mimicking specific individuals within the population.

VI. Conclusion

Although domain experts are an important part of the modeling process, in cases where it is possible to obtain data, it is desirable to reduce some of the subjectivity in parameter selection. Our initialization method of combining agent-based models with survey data allows us to streamline model creation, making the process more automatic. In this paper, we have presented the aspects of our campus modeling effort that apply to the widest possible range of urban microsimulations.

One simple improvement that we are planning to make in the future is to add faculty/staff into our simulation; this was not a priority initially since previous work has shown that faculty/staff activity profiles have a much lower entropy and are inherently easier to predict than student profiles [5]. Supplementing the simulation with additional information about semester class scheduling is likely to yield the largest forecasting improvement at the cost of making the simulation less

applicable to other urban modeling problems. A large amount of class attendance and scheduling information is collected by the university and could be added to the simulation without requiring additional survey efforts.

ACKNOWLEDGMENTS

This research was supported by DARPA award N10AP20027 and AFOSR YIP FA9550-09-1-0525.

REFERENCES

[1] C. M. Macal and M. J. North, "Agent-based modeling and simulation: desktop ABMS," in *Proceedings of the Conference on Winter Simulation*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 95–106.

[2] C. Andrieu, N. de Freitas, A. Doucet, and M. Jourdan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, pp. 5–43, 2003.

[3] J. Oakes, "Invited commentary: Rescuing Robinson Crusoe," *American Journal of Epidemiology*, vol. 8, no. 1, pp. 9–12, 2008.

[4] I. Benenson, P. Torrens, W. Europe, and J. Portugali, "Geosimulation: automata-based modeling of urban phenomena," *Environment and Planning B: Planning and Design*, vol. 31, no. 4, pp. 589–613, 2004.

[5] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Pervasive and Ubiquitious Computing*, vol. 10, pp. 255–368, 2006.

[6] J. Letchner, J. Krumm, and E. Horvitz, "Trip router with individualized preferences (TRIP): Incorporating personalization into route planning," in *Proceedings of Conference on Innovative Applications of Artificial Intelligence*, 2006.

[7] U. Wilensky, 1999, NetLogo. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. Retrieved from: http://ccl.northwestern.edu/netlogo/.

[8] R. Allan, "Survey of agent based modelling and simulation tools," Science and Technology Facilities Council, Tech. Rep. DL-TR-2010-007, 2010.

[9] R. Liu, J. Tao, N. Shi, and X. He, "Bayesian analysis of the patterns of biological susceptibility via reversible jump MCMC sampling," *Computational Statistics & Data Analysis*, vol. 55, no. 3, pp. 1498–1508, 2011.

[10] Y. Liu, Q. Wang, J. Liu, and T. Wark, "MCMC-based indoor localization with a smart phone and sparse wifi access points," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, March 2012, pp. 247–252.

[11] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[12] M. Laine, 2012, MCMC toolbox for Matlab, Finnish meteorological institute. Retrieved from: http://helios.fmi.fi/ lainema/mcmc/.

[13] X. Jin and L. Jie, "A study of multi-agent based models for urban intelligent transport systems," *International Journal of Advancements in Computing Technology*, vol. 4, no. 6, pp. 126–134, April 2012.

[14] R. Jordan, M. Birkin, and A. Evans, "Agent-based modelling of residential mobility, housing choice and regeneration," in *Agent-Based Models of Geographical Systems*. Springer Netherlands, 2012, pp. 511–524.

[15] T. A. Kohler, R. K. Bocinsky, D. Cockburn, S. A. Crabtree, M. D. Varien, K. E. Kolm, S. Smith, S. G. Ortman, and Z. Kobti, "Modelling prehispanic Pueblo societies in their ecosystems," *Ecological Modelling*, vol. 241, pp. 30–41, 2012.

[16] D. Brown, R. Riolo, D. Robinson, M. North, and W. Rand, "Spatial process and data models: Toward integration of agent-based models and gis," *Journal of Geographical Systems*, vol. 7, no. 1, pp. 25–47, 2005.

[17] H. Gimblett, *Integrating geographic information systems and agent-based modeling techniques for simulating social and ecological processes*. Oxford University Press, USA, 2002.

[18] X. Jin and R. White, "An agent-based model of the influence of neighbourhood design on daily trip patterns," *Computers, Environment and Urban Systems*, 2012.

[19] H. Dia, "An agent-based approach to modelling driver route choice behaviour under the influence of real-time information," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5-6, pp. 331–349, 2002.

[20] N. Schwarz and A. Ernst, "Agent-based modeling of the diffusion of environmental innovations–An empirical approach," *Technological forecasting and social change*, vol. 76, no. 4, pp. 497–511, 2009.

[21] B. Chen and H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 485–497, June 2010.

AmalgaCloud: Social Network Adaptation for Human and Computational Agent Team Formation

Kirill Osipov-Lvoff
Department of EECS
University of Central Florida
Orlando, Florida
Email: karl@knights.ucf.edu

Gita Sukthankar
Department of EECS
University of Central Florida
Orlando, Florida
Email: gitars@eecs.ucf.edu

***Abstract*—Many complex problems can be solved through an effective organization of human experts and software agents (services) connected by a social network where each node contributes a unique skill set needed to enable a higher order problem solving capability of the group. Recent work in crowdsourcing applications based on enterprise social networks (e.g. PeopleCloud) has shown that the group problem solving approach can be extended to enterprise and potentially Internet-wide scales. However, systems operating at such scales assume that candidate group participants make decisions about which groups to join based on limited connectivity and local information.**

This paper focuses on the relationship between network adaptation for candidate group participants and performance of problem solving groups. We demonstrate that systems that expect to form groups (e.g. crowdsourcing) by engaging participants equipped with diverse skill sets require more sophisticated network adaptation strategies than what can be expected based on previous research. To address this need, we evaluate a set of network adaptation algorithms for crowdsourcing and present some empirical results from a simulation based study.

I. Background

Problem solving activities are increasingly based on self-organized groups (communities or teams) that collaborate across functions, divisions and levels of their respective organizations [1] and team formation is growing in importance as a business problem [2]. In the area of human team organization, the operations research (OR) field developed several integer linear program formulations specifying optimal team composition which can be solved using branch-and-cut [3], approximation heuristics [4], genetic algorithms [5] or simulated annealing [6]. Advances in social graph data mining coupled with traditional OR techniques enabled novel solutions to the problem of human team formation [7]. Team formation supported by social network adaptation has been shown to increase team [8] and organizational performance [9] [10].

Crowdsourcing [11] based approaches emphasize the role of human experts in problem solving groups. Systems such as PeopleCloud [12] [13] help build specialized "crowds" (teams) of information technology (IT) experts for purposes of IT service delivery. In context of these systems, crowdsourcing provides a convenient abstraction, hiding the issues related to integration of the human expert (IT administrator) to a computational system (e.g. server monitoring agent) by treating the integration as one of the skills in the human expertise skill set.

Organization of independent computational agents into Multi-Agent Systems (MAS) or agent teams has been proposed as an alternative to monolithic agent-based software systems and as an approach for addressing bounded rationality limitations [14]. While MAS researchers focused on use of computational agents in a support or assistant role to humans, other researchers have proposed models where human and computational agents (the latter implemented as software services) jointly participate in problem solving activities, for example in mixed systems of humans and software services [15], social compute units [16] and human-provided application stores [17].

Some researchers have also proposed to construct social networks that exclude human participants and enable systems based solely on computational agents (implemented as web services) to form groups (composites) capable of solving higher order problems [18].

II. AmalgaCloud Introduction and Overview

This paper reports on results of experiments used to guide design of AmalgaCloud [19], a research project by the authors to prototype an internet service for organizing problem solving teams from a social network of human and computational agents on a basis of a structured problem definition. An agent in AmalgaCloud has expertise (skills) and can form or join teams with agents having complementary expertise from its social network. The social network has a continuously changing structure as agents use alternative strategies to modify network connectivity in a search to improve their problem solving performance. We explore alternative network adaptation (modification) strategies and their relationship to the AmalgaCloud problem solving performance to better inform AmalgaCloud design.

Our contributions.

- We describe an extension of the team formation algorithm studied by [7] to a parallel setting where multiple, independent agents self-initiate team formation proposals and choose between alternative proposal variations to maximize their team formation performance.

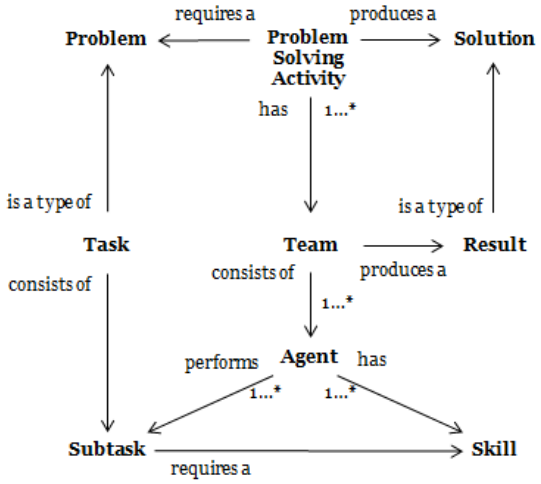


Fig. 1. AmalgaCloud is a research project by the authors to prototype an internet service for organizing problem solving teams from a social network of human and computational agents on a basis of a structured problem definition. The figure is relationship diagram for AmalgaCloud concepts, clarifying the terminology and the relationships between terms. The arrow labels should be read in the direction of the arrow, e.g. Task is a type of a Problem. The arrow represent a one-to-one relationship, unless one-to-many relationship is specified using the 1...* notation. One-to-many notation should be read in the direction of the arrow e.g. one Agent has many Skill(s).

- We provide simulation based evidence in support of network adaptation policies that take into consideration agent's knowledge of its performance on task completion (performance-based strategies). We also show that commonly used network adaptation policies based on preferential attachment (structure-based strategies) should not be assumed as effective for team formation that relies on agents with multiple skill sets.
- We report measurements of comparing both structure and performance based network adaptation in conjunction with the extension. The measurements are interpreted and presented as design guidelines for the AmalgaCloud.

Roadmap.

The rest of this paper is organized as follows: in the remainder of Section II we review salient features of AmalgaCloud and outline issues and challenges faced by the authors in applying existing research results to AmalgaCloud design. We focus on scalability of the exiting research results relative to increase in the number of unique skills per agent and measure whether the structural and performance based network adaptation strategies proposed by [9] [10] remain effective in presence of exponential increase in the space of potential skills configurations. In Section III we provide a formal definition of the team formation and network adaptation models. In Section IV we present the experimental approach to evaluation of alternative network adaptation strategies. We review related work about problem solving with systems of agents in Section VI and discuss our results in Section V. We conclude in Section VIII.

A. AmalgaCloud

As a detailed technical description of AmalgaCloud [19] is outside the scope of this paper, we provide highlights of the salient features of the system design. A formal, mathematical treatment of the models for tasks, skills and agents will be provided in the later sections of the paper.

Structured problem definition. AmalgaCloud system is designed to support a restricted category of problems that exist outside of the system boundaries, are specified in a predefined structured format and can be solved by multiple teams engaged concurrently and independently in problem solving activities for a restricted period of time to arrive to a solution. In formal specification for AmalgaCloud, we require that problems are:

- *finite*, a problem must be solvable within a finite time internal which is given prior to the start of the problem solving activity.
- *scoped*, a problem must have a limited scope in terms of the maximum number of agents that must participate in the problem solving activity to produce a problem solution.
- *verifiable*, a problem must have an observer (or observers) to accept or reject a proposed solution; a rejected proposal is considered not to be a solution to a problem. In a given problem formulation an observer agent be may required. Alternatively, the accept or reject decision may be performed by a shared observer agent.
- *isolated*, a problem must be solvable by multiple, isolated and concurrent problem solving activities.

In this paper, we will use "task" to describe a problem that has these properties. The structured problem definition describes each task as consisting of multiple subtasks. A subtask can be performed (completed) by applying a specific skill, i.e. there is a one to one relationship between a subtask and a skill. A result is a completion of all subtasks for the corresponding task. An illustration of these concepts is provided on Figure 1.

Problem properties described above include a broad category of IT tasks related to service delivery and troubleshooting in complex IT environments [12] [13]. For example, a task many involve a team of experts with skills in networking, hardware, operating systems and application software working together to identify the root cause of a decline in application performance. The isolated property of the problem ensures that multiple teams can produce independent (i.e. created concurrently and in isolation) results. The results can be aggregated to identify the most frequently reported or the most likely root cause of a problem.

Other aspects of the structured problem definition are covered in more detail later in this section.

Problem solving agents. AmalgaCloud assumes the existence of a social network (graph) where vertices represent skill (expertise) profiles of human or computational agents. For example, for human profiles the skills may include "reinstall operating system" while for a computational agent the skill may be "report on an operating system memory utilization".

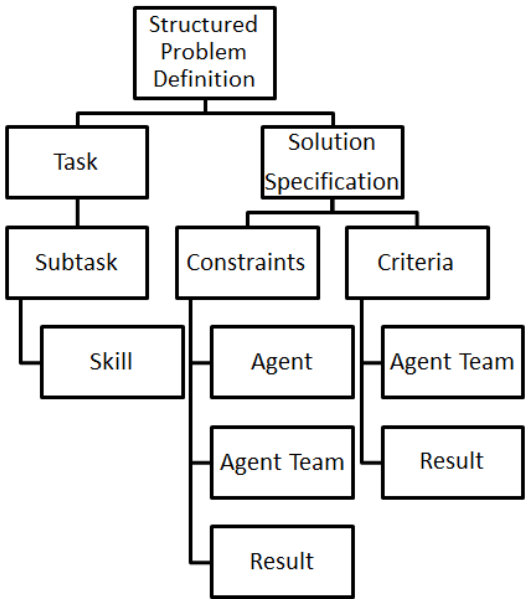


Fig. 2. Structured problem definition (SPD) introduced earlier in this section includes information about the task to be performed, about the agents that must perform the task and about the result. The solution specification part of the SPD, consists of 1) constraints, which are inclusion/exclusion rules for agents, agent teams and the result; and 2) criteria, which are rules (objective functions) for specifying preferences over the space of potential solutions. For example, an agent team constraint may exclude any team formation solutions that have less than three agents on a team; a criteria may specify that a team formation solution should have a team with as few members as possible, as long as the team has all the specified skills for a team. The result part of the solution specification allows for additional restrictions such as ensuring that a team can produce a result within a specified period of time (constraint) or within as short of a time period as possible (criteria).

Existence of an edge between any two agents represents bidirectional awareness between the agents. The edge may optionally have additional relational information associated with the agents, for example communication cost or records of past interactions.

Team formation. A formal description of the team formation problem will be provided later in this paper. Intuitively, team formation describes a process for selecting a group of agents from the social network such that the team formed by the identified agents meets one or more requirements, which must include having the capability to perform the task specified in the problem definition. Other requirements may be derived from the information specified in the social graph edges between the members, for example communication cost, history of past interactions or presence of common social network neighbors. During team formation, the agents are only considering joining teams that are in the agent immediate (directly connected) network neighborhood.

The problem solving activity interactions that take place in a mixed team of computational and human agents following team formation are outside the scope of AmalgaCloud; [15] describes how interactions can be implemented in mixed systems.

Network adaptation. The lifecycle of the AmalgaCloud platform consists of team formation events, sequential or concurrent. Network adaptation refers to agent initiated change in the social network in the time between the events enabling the agents to prepare to future problems and future team formation events. To represent limited attention or workload capacity, agents have an upper limit on the number of possible connections to other agents in the network.

Design considerations. We focus on problem solving systems that are designed to be open and distributed, defined by [20] as ones where the structure of system itself is capable of dynamically adapting given a problem and where system components (human or computational agents) are not known in advance. While in MAS [21] information about skill sets for agents is known at application design time, our prototype is more similar to PeopleCloud [12], in that we can collect information about both the task and the skill sets of its social network participants at system runtime. Information about the participants can be collected dynamically from external social network data sources and is processed against constraints (e.g. history of participant contributions, knowledge expertise, participation levels) to identify the right agents for a given task.

Existing research shows that there exists the need for a team formation platform that can support a range of alternative optimization formulations [7] [22] [23] [24] [25]. In AmalgaCloud, we extend the structured problem definition (SPD) to include a collection of constraints and criteria that collectively restrict and rank possible solutions to the team formation problem. The solution specification part of the SPD, consists of 1) constraints, which are inclusion/exclusion rules for agents, agent teams and the result; and 2) criteria, which are rules (objective functions) for specifying preferences over the space of potential solutions. For example, an agent team constraint may exclude any team formation solutions that have less than three agents on a team; a criteria may specify that a team formation solution should have a team with as few members as possible, as long as the team has all the specified skills for a team. The result part of the solution specification allows for additional restrictions such as ensuring that a team can produce a result within a specified period of time (constraint) or within as short of a time period as possible (criteria).

There exists a broad range of research on how to use centralized matchmaking agents for solving problems by relying on multiple problem solving groups working in parallel [21]. In MAS, identifying the right team for a given task (team formation) can be performed by using a specialized matchmaker (interface) agent [26] or through peer to peer sharing of goals and plans across agents [27]. In contrast, team formation algorithms described in [7] and extensions [23] [24] [22] [25] [28] rely on a single, central entity with a global view of the social network of candidate team members. It is unclear whether the centralized techniques can apply at levels of scale and dynamicity in systems intended to service large enterprises or the entire population of the World Wide Web.

Decentralized, peer to peer based approaches for team formation have greater potential to scale to larger pools of candidate team participants. Some of the MAS solutions have relied on peer to peer based team formation, using plan and goal sharing across computational agents [26]. Some studies focusing specifically on team formation [9] [10] [29] have quantitatively evaluated the relationship between alternative team collaboration network structures and overall effectiveness of groups in solving problems or performing tasks. These studies examine team formation dynamics, i.e. the changes in collaboration network connectivity over time as collaborators seek to leave and join possible groups in order to improve both their local and system-wide problem solving performance. While the peer to peer based approaches to team formation have demonstrated greater scalability, the agents are faced with the problem of decision making on the basis of limited information about the agents within their local connection vicinity.

It is desirable to specify skills at a sufficiently fine grained detail level so as to avoid ambiguity. In addition, the skill set may grow at system runtime as participants list or acquire additional skills. However, the increase of the participant skill set leads to an exponential growth in the number of the potential groups where the participant may be included[1].

III. Model Formulation

In our simulation model, there is a population of N agents represented by set $A = \{a_1, a_2, ..., a_N\}$. Each agent is connected to a portion of the agent population via a social network which is modeled using a symmetric adjacency matrix E, where an element of the matrix $e_{ij} = 1$ indicates an undirected edge between agents a_i and a_j. In the paper we distinguish between *first order neighbors* of a_i defined as $N_i^1 = \{a_j : e_{ij} = 1, j \neq i\}$ and *second order neighbors*, defined as $N_i^2 = \{a_k : e_{ij} = 1, e_{jk} = 1, e_{ik} = 0, k \neq i\}$. The degree of an agent a_i is defined as d_i. Each agent a_i has a set of skills S_i which is a subset of size S_A sampled randomly from a uniform distribution over a set of the universe of skills S; in the previous work such as [6] [7], the assumption is that each agent only possesses a single skill so $S_A = 1$. The agents interact over multiple time steps in a simulation. At every time step ts of the simulation, a single task T_{ts} is randomly generated by sampling without replacement from a uniform random distribution of the set of skills S to generate a set of predefined size T_A. A set of agents (group) $G = \{a_k, ..., a_l\}$ is said to be capable of executing a task T iff $T \subset S_k \cup ... \cup S_l$. In all of the simulations described in this paper, $S_A < T_A$ so that more than one agent is required to execute any given task. Every T_{ts} is broadcast to all N agents and more than one group of agents may have the skills (potential) needed to execute T. As described in more detail later in this paper, the quantity of these potential groups is one of the key metrics for evaluation of alternative network adaptation algorithms.

The model avoids concurrency issues by merging the steps related to formation of a team and execution of a task into a single time step of the simulation. Once all of the potential groups are identified, every agent commits to (joins) one of the groups using the algorithm described later in the paper. As long as a team has enough committed agents capable of executing T_{ts} then the team is considered to have executed T_{ts} at the conclusion of the ts step. Given the focus of this paper on Internet scale team formation, this formulation permits the possibility that multiple groups may complete the task in parallel. While it is possible to introduce coordination or election mechanisms to ensure that a task is performed only once, this topic is outside the scope of this paper.

A. Initial Social Network Connectivity

To establish connectivity, all agents are randomly assigned a position on a square grid with side of size $\sqrt{N}$ (based on the N value from Table I). Distance between the agents is measured under an assumption of toroid connectivity between grid's edges using Manhattan distance measure, D_{ij}. For every agent a_i, an undirected edge e_{ij} is established to every other agent a_j, as long as D_{ij} is less than or equal to a predefined constant D (see Table I). For every agent a_i, the initial N_i^1 connectivity configuration as explained here is identical for all simulation scenarios described in this paper. The algorithm implementing this connectivity configuration is identical to the random geometrical graph generation approach followed by [29] [10].

B. Candidate Group Selection and Group Formation

Since agents are operating on the basis of limited information about potential groups, every agent must make a decision about which groups can solve the tasks and which groups the agent should join to execute the task. At every time step of the simulation, an agent a_i knows only which skills are available to agents in its first order social network neighborhood N_i^1. Before joining or forming a team, an agent a_i must compute $G_{i,ts}$ which is a set of potential groups that have the skills needed to execute T_{ts}. Agent a_i computes the intersection of the sets S_i and T and whenever the intersection of the two sets is non-empty (i.e. the agent has at least one skill needed for the task), attempts to perform the computation of a set of candidate groups that the agent expects to execute the task. The computation problem is equivalent to an optimization by minimizing the sum of indicator functions which specify whether an agent $a_j \in N_i^1$ should be in the potential team set:

$$minimize \sum_{j=1}^{N} I_{G_{i,ts}}(a_j) \tag{1}$$

subject to the constraint

$$T_{ts} \subset \bigcup_{k \in G_{i,ts}} S_k \tag{2}$$

[1]Consider a social network where a vertex (node) represents an agent and the vertex degree d stands for agent's awareness of d other agents in the network, each edge representing a potential collaboration. Under the assumption that the agent represented by the node has S_A skills, the agent may offer any of its $2^{S_A} - 1$ subsets of skills to a collaborator. Considering all of its d potential collaborators, the agent may be a candidate for at most $d(2^{S_A} - 1)$ collaborative groups.

The equations 1 and 2 are an example of the well known minimum set covering problem [30] ($MinSetCover$). In this instance, the objective is to find the smallest possible set of agents such that their respective skills S_j "cover" the skills required for a given task T. While set covering is a classic example of an NP-complete problem [31], it has well known approximate solutions [32] and in context of bounded social network neighborhoods can be solved exactly and efficiently using industrial scale solvers [33] .

An agent can both initiate its own team and be invited to participate in a team initiated by another agent. Given any two agents a_i and a_j, a_i could be a member of $G_{i,ts}$ or $G_{j,ts}$. We will refer to groups in $G_{i,ts}$ where a_i is a member as self-initiated and denote them by $SG_{i,ts}$. Other-initiated groups $OG_{j,ts}$, are those where a_i is a member of $G_{j,ts}$. As the union of $SG_{i,ts}$ and those groups in $OG_{j,ts}$ with a_i as a member may contain members of N_i^2; given $g_i, g_j \in G$ agents follow a preference policy $Pref(g_i, g_j)$ to select which team to join. The policy is formulated to ensure that agents encourage smaller groups which maximizes the number of groups that can execute a task and within a single time step, prefer groups that have as many agents from the first order neighborhood as possible, which increases the importance of an effective network adaptation strategy. To decide the team to join, the agent performs additional filtering and preference sorting on all of its candidate groups based on a policy where the agent prefers

- smaller groups over larger groups
- groups most similar to its own team proposal
- groups most similar to its first order neighborhood

The preference for smaller teams increases the total number of open positions for agents and consequently the total number of candidate teams. Since an agent always proposes teams based on its first order neighborhood, the remaining two preferences ensure that an agent chooses a team for which it has the maximum amount of information available through its social network.

The policy is defined formally as:

$$Pref(g_i, g_j) = \begin{cases} g_i & |g_i| < |g_j| \quad (3) \\ g_i & |G_i \cap g_i| > |G_i \cap g_j| \quad (4) \\ & \wedge |g_i| = |g_j| \\ g_i & |g_i \cap N_i^1| > |g_j \cap N_i^1| \quad (5) \\ & \wedge |G_i \cap g_i| = |G_i \cap g_j| \\ & \wedge |g_i| = |g_j| \\ g_j & \text{otherwise} \quad (6) \end{cases}$$

The procedure followed by the agents in forming a collection of groups to execute a given task is summarized in Algorithm 1 listing.

Algorithm 1 Agent Group Selection and Participation Algorithm

```
procedure SELECT-JOIN-GROUP(T_ts, A, N)
    for all i ∈ {1...N} do                    ▷ iterate over agent index
        if |S_i ∩ T_ts| > 0 then              ▷ run asynchronously
            SG_{i,ts} ← MinSetCover(N_i^1, T_ts)
        end if
    end for
    for all i ∈ {1...N} do
        OG_{i,ts} ← ⋃_{j∈{1...N}} {SG_{j,ts} : a_i ∈ SG_{j,ts},
i ≠ j}
        g_{i,ts} ← Join(SG_{i,ts} ∪ OG_{i,ts}, Pref)
        G_ts ← G_ts ∪ {g_{i,ts}}
    end for
    return G_ts
end procedure
procedure JOIN(G, ComparePolicy)
    G_sorted ← ComparisonSort(G, ComparePolicy)
    g ← Head(G_sorted)   ▷ first element of the sorted list
    return g                                  ▷ discard the tail of the list
end procedure
```

C. Node Degree Based Network Adaptation

Following task execution and prior to the conclusion of every time step of the simulation, agents have an option to adapt their network connectivity. Agent a_i examines a set of agents N_i^2 such that a_k is a member of N_i^2 iff all of the following are true

- there exists an agent a_j such that it is connected to a_i via e_{ij}
- a_j is connected to a_k via e_{jk}
- a_k is not connected to a_i via e_{ij}
- a_i and a_k are not the same

Intuitively, the formal description above specifies the set of all second-order neighbors (i.e. neighbors of a_i's neighbors) with exception of those that are also connected to a_i directly. Using the node degree of the agents in A, a_i defines two probability mass functions:

$$P(a = a_i) = \frac{1}{|\{A\}|} \quad (7)$$

and

$$P(a = a_j) = \frac{d_j}{\sum_{a_k \in N_i^2} d_k} \quad (8)$$

By sampling from the probability density function 7, a_i selects an agent a_j and deletes e_{ij} thus removing a random first order neighbor. Next, a_i samples from probability density function 8 to select agent a_k and creates e_{ik} edge. This approach is designed to implement the preferential attachment policy described in [29].

Our model also enables a complementary policy based on use of an agent's local team formation performance for the network adaptation decision. Since we measure agent performance in terms of its effectiveness in joining groups and completing associated tasks, the agent performance based network adaptation policy can be summarized as follows: if at time time step ts an agent a_i does not select and join a team (Algorithm 1), then prior to start of $ts + 1$, a_i will adapt it's

network connectivity using the preferential policy approach described in this section. This performance policy is based on the expectation that if a subset of agents A_H has created a high performing network connectivity configuration$N1_H$, then their $N1_H$ should remain static across time steps while poorly performing agents should adapt their set of $N1_P$ in an attempt to improve their performance.

Earlier research [9] [10] has addressed the question of an appropriate strategy for selection of the candidate agents for network adaptation as well as selection criteria for identifying connection destinations for the candidate agents. There is evidence that strategies that enable agents to adapt connectivity based on local structural information outperform strategies where agents attempt to model global network performance [10]. Consequently we have not attempted to incorporate global variables (e.g. total number of groups formed) into individual agent decisions about network adaptation.

IV. Method

In the evaluation of the model, we compare implications of alternative connectivity dynamics on team formation in presence of a variable number of skills that every agent can contribute to a team. This section provides an overview of give related scenarios that have been simulated as part of the study, each scenario focusing on a unique set of network adaptation configurations and strategies. Each of the scenarios has been simulated using a combination of common and scenario specific sets of parameters described in Table I. For all of the scenarios we have collected the same set of measurements to compare the impact of skill set diversity in team formation to static and preferential node attachment connectivity models.

The first scenario (S1 / Single Skill, No Network Adaptation) assumes the absence of any connectivity changes relative to the initial, randomly created agent network. To reproduce results from [10] we also restrict the number of skills per agent with $S_A = 1$. S1 serves as a baseline to showcase average performance of the single skill agent network across multiple simulations with a nontrivial sample set of possible initial random geometric graph configurations.

The second scenario (S2 / Single Skill, Structure Based Network Adaptation), follows [29] to reproduce the node degree based network adaptation under the $S_A = 1$ assumption. The first step of every simulation under this scenario assumes random geometric graph connectivity described in section III-A. Prior to the beginning of the second and prior to every subsequent step of the simulation in this scenario, every agent a_i updates its set of edges as described in the section III-C on adapting network connectivity using the preferential policy approach.

The third scenario (S3 / Diverse Skills, No Network Adaptation) extends S1 through introduction of diverse agent skill sets described earlier in this paper. No network adaptation is performed in this scenario as it is designed to serve as as an illustration of the impact of a larger number of skills in a system on the team candidate identification and team formation performance.

TABLE I
Simulation Scenario Parameters

Name	Value					Description
	Scenario					
	S1	S2	S3	S4	S5	
NumSimulations	256					Number of times each scenario has been simulated.
NumSteps	128					Number of time steps per scenario, also number of tasks randomly generated per simulation.
N	64					Number of agents in the simulation scenario social network
D	2					Manhattan distance for initial random geometric graph agent connectivity such that a_i is connected to all neighbors a_j that have $D_{ij} \leq D$
$\|S\|$	16					Number of distinct skills in the simulation scenario
T_A	8					Total number of distinct, randomly chosen skills per task
S_A	1	1	4	4	4	Total number of distinct, randomly chosen skills per agent

The fourth (S4 / Diverse Skills, Performance Based Network Adaptation) and fifth (S5 / Diverse Skills, Structure Based Network Adaptation) scenarios both extend S3 using alternative network adaptation strategies described in the section III-C. S4 uses the agent performance based network adaptation policy while S5 relies on preferential network attachment network adaptation for every agent in the network after every time step. The latter scenario's approach is designed to illustrate performance of a purely random mechanism which does not incorporate consideration of the overall system performance. Introduction of this scenario is motivated by [10] which demonstrates that structural policies may outperform adaptation strategies involving agent estimation of system performance based on locally available information.

Since the results of the variations of the network adaptation policy on the network structure are similar across scenarios S2, S4 and S5, they are illustrated with the example shown on Figure 3. As a consequence of the configuration parameters from Table I, every agent a_i is instantiated (at $t = 0$) with $d_i = 16$. The figure shows that the network adaptation policy progressively modifies the node degree distribution towards a concentration of high degree nodes with "fat tails" of single digit degree nodes. The degree distribution does not change significantly beyond $t = 30$ to warrant additional illustrations. In the figure, every distribution is paired with a corresponding illustration of the underlying graph providing

TABLE II
GROUP FORMATION RESULTS

Scenario	Groups Formed				Candidate Groups			
	μ	σ	M	m	μ	σ	M	m
S1	0.0054	0.0731	1	0	0.0008	0.0295	3	0
S2	0.0852	0.2884	2	0	0.0216	0.1853	6	0
S3	3.1067	1.2684	8	0	0.8349	1.0439	7	0
S4	2.9661	1.2452	8	0	0.6705	1.3598	8	0
S5	2.8752	1.2012	8	0	0.6271	0.9219	8	0

a representation of the network adaptation algorithm effect on the graph structure.

Every scenario is studied through execution of $NumSimulations = 256$ simulation sessions, each session consisting of a fixed and equal number of steps for all simulations. For every simulation step we measure the number of candidate groups identified for every agent as well as the number of groups formed during the step. In addition to tracking the descriptive statistics (mean, standard deviation, and minimum, maximum) for these variables on per step basis, we also compute statistics of these variables across all steps in a simulation and for all of the simulations in a scenario. The scenario scope measurements are needed to reduce potential bias due to selection of random geometric connectivity graphs as initial conditions in the first step of every simulation. Simulation results are summarized in Table II [1].

V. DISCUSSION

When comparing the results for S1 and S2, we confirm the observation that the use of agent performance based preferential node attachment policy (in S2) leads to both a higher number of groups formed and of candidate groups where the agents could participate. Compared to [29] [10] the mean values of both variables are lower due to the difference between the total number of skills in the simulations. As argued by earlier research, use of structural, preferential policy based network adaptation leads to significantly better performance in these scenarios resulting in over an order of magnitude improvement in the number of groups formed (from 0.0054 to 0.0852) and number of candidate groups per agent (from 0.0008 to 0.0216).

As argued earlier in this paper (footnote 1), introduction of a larger number of skills per agent can exponentially increase the number of potential groups where the agent can participate. As shown by the results for S3, even without any network adaptation, increase in skill set cardinality leads to an increase of approximately three orders of magnitude (from 0.0008 to 0.8349) for the mean number of the candidate groups per agent with a parallel increase (from 0.0054 to 3.1067) in the mean number of groups formed across the simulations in the scenario.

Given the results from research on network adaptation and evidence from S2, one may expect further improvement to the S3 results through the introduction of preferential attachment policy under the assumption of multiple skills per agent. However results of the simulation demonstrate that not to be the case. In S4, the mean number of groups formed across simulations (2.9661) is not statistically significantly different from the same number in absence of the policy, while the number of the candidate groups per agent measurably dropped (from 0.834846 to 0.6705). Further, S5 demonstrates that use of agent performance information for network adaptation is not measurably different from using a purely random preferential attachment policy.

Note from Figure 3 that over the course of the simulation, an increase of the average weighted degree (from 16 to 24.891) of the social network was less than by a factor of two. One interpretation of this result suggests that if the number of skills in the system $|S|$ increases linearly while the number skills per agent S_A stays constant, the preferential node attachment network adaptation policy becomes less effective as the total space of possible task assignments grows exponentially.

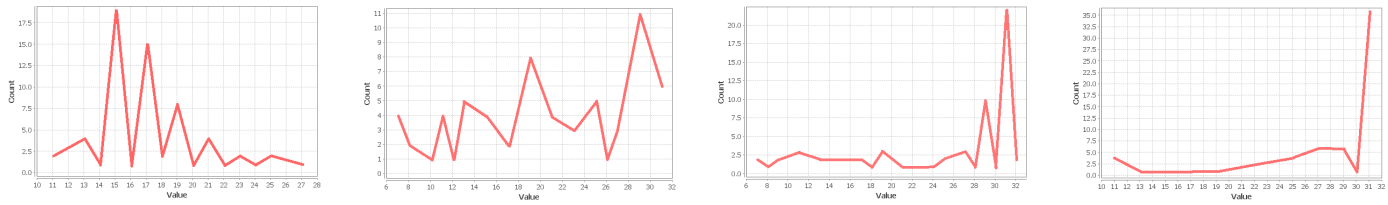
VI. RELATED WORK

The area of community discovery (detection) is complementary to our research. Community detection work as advanced by [34] [35] relies on a record of past (relative to the time of the community discovery query) interactions among nodes in a network. For example, by comparing frequency of edge distribution in a network to random edge distribution it is possible to measure modularity in a network and thus discover communities. Our research does not require historical data on node interaction as we study how to form teams that may not have previously existed as a community. However, our approach can benefit from community discovery as history of past interactions can positively inform team formation.

In the solution to a TEAM FORMATION problem, Lappas et al [7] proposed to model inter-dependencies between agents using an undirected, weighted social graph. Edge weights in the graph can incorporate measures such as effectiveness of agent-to-agent communication when grouping together agents into teams. The specific algorithm in the paper finds a team of skilled individuals which minimizes communication cost among members of the team. However [7] assumes a static graph structure in answering the team formation problem and is missing a prescriptive model for setting edge weights in a way independent of the application domain.

Li and Shan [23] extend [7] to account for the tasks where some sub-tasks (a sub-problem) must be performed by a specific number of skilled individuals. Yin et al [24] extend [7] with a diversity metric based on measurements of influence that potential team members receive from their peers in neighboring graph nodes. The metric ensures that the team formation solutions are biased towards having members that influence each other as little as possible. Anagnostopoulos and Becchetti [36] describe a TASK ASSIGNMENT problem which seeks to ensure a balanced workload across team members, minimizing maximum load over all the experts and also provide an extension [22] to include communication costs

[1]The values in the table represent mean (μ), standard deviation(σ), maximum (M) and minimum (m) across all $NumSimulations$ for a scenario



(a) AWD = 17.266, t=0 (b) AWD = 20.484, t=10 (c) AWD = 24.359, t=20 (d) AWD = 27.484, t=30

Fig. 3. Simulation results. Subfigures a-d show the frequency distribution of node degrees for time steps 0 through 30 along with specific values for average weighted degree (AWD).

similar to [7]. Kargar and An [28] point out the a minimum spanning tree (MST) based communication cost function used by [7] does not effectively model team formation scenarios where individual team members have to communicate with each other directly. Also Kargar and An [28] provide an alternative communication cost functions that results in more stable (relative to minor communications cost graph changes) solutions than those suggested by [7].

VII. FUTURE WORK

The simulation based study in this paper does not provide a detailed, analytical treatment of the relationship between the network adaptation policies and the system-wide performance. Future research should focus on further simplification of the model described in this paper to identify the key factors negatively impacting scalability of the network adaptation policy.

The implementation described in this paper relied on a simplified solver for the minimum set cover algorithm. Follow on work will integrate our simulation with a production solver (e.g. CPLEX) to study the model with larger scale data.

VIII. CONCLUSION

An increasing number of systems seek to exploit information available in Internet scale social networks to identify teams of experts for knowledge discovery [13] or for task-oriented crowdsourcing [12]. When considering performance of such systems in terms of their ability to organize groups, it is reasonable to expect that results from studies on group formation [29] [10] should extend to more sophisticated models of individual agents and their contributions to potential groups. Our simulation based study demonstrates that models of agent capabilities that allow for runtime changes to agent skill sets (for example in crowdsourcing systems like PeopleCloud [12]) introduce scaling difficulties for traditional network adaptation policies based on preferential node attachment.

We have shown that use of more detailed skill set descriptions per agent (i.e. in terms of a number of skills per agent) is desirable as it motivated by potential crowdsourcing applications [13] and has a net positive effect on the number of the candidate groups where an agent can contribute its skills and the total number of groups that can be formed by a system. However further research is needed to more precisely analyze and quantify the impact of preferential attachment policies, and to research alternative network adaptation strategies.

REFERENCES

[1] P. Adler and C. Heckscher, "Collaborative Community," *Ask Magazine*, vol. 23, pp. 41–45, 2006.

[2] F. Bonchi, C. Castillo, and A. Gionis, "Social Network Analysis and Mining for Business Applications," vol. 2, no. 3, pp. 1–37, 2011.

[3] A. Zzkarian and A. Kusiak, "Forming teams: an analytical approach," *IIE Transactions*, vol. 31, pp. 85–97, 1999, 10.1023/A:1007580823003. [Online]. Available: http://dx.doi.org/10.1023/A:1007580823003

[4] E. L. Fitzpatrick and R. G. Askin, "Forming effective worker teams with multi-functional skill requirements," *Comput. Ind. Eng.*, vol. 48, no. 3, pp. 593–608, May 2005. [Online]. Available: http://dx.doi.org/10.1016/j.cie.2004.12.014

[5] H. Wi, S. Oh, J. Mun, and M. Jung, "A team formation model based on knowledge and collaboration," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9121–9134, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2008.12.031

[6] A. Baykasoglu, T. Dereli, and S. Das, "Project team selection using fuzzy optimization approach," *Cybern. Syst.*, vol. 38, no. 2, pp. 155–185, Feb. 2007. [Online]. Available: http://dx.doi.org/10.1080/01969720601139041

[7] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, p. 467, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1557019.1557074

[8] N. Glance and B. Huberman, "Organizational fluidity and sustainable cooperation," in *From Reaction to Cognition*, ser. Lecture Notes in Computer Science, C. Castelfranchi and J.-P. Mller, Eds. Springer Berlin / Heidelberg, 1995, vol. 957, pp. 89–103, 10.1007/BFb0027058. [Online]. Available: http://dx.doi.org/10.1007/BFb0027058

[9] M. Gaston and J. Simmons, "Adapting network structure for efficient team formation," *Proceedings of the AAAI 2004 Fall*, 2004.

[10] M. E. Gaston and M. DesJardins, "Agent-organized networks for dynamic team formation," *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems - AAMAS '05*, p. 230, 2005.

[11] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, ser. Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, 2010. [Online]. Available: http://books.google.com/books?id=atfCl2agdi8C

[12] M. Lopez, M. Vukovic, and J. Laredo, "PeopleCloud Service for Enterprise Crowdsourcing," *2010 IEEE International Conference on Services Computing*, pp. 538–545, Jul. 2010.

[13] P. Ypodimatopoulos, M. Vukovic, J. Laredo, and S. Rajagopal, "Server Hunt: Using Enterprise Social Networks for Knowledge Discovery in IT Inventory Management," *2011 Annual SRII Global Conference*, pp. 418–425, Mar. 2011.

[14] K. Sycara, "Multiagent systems," *AI Magazine*, vol. 19, no. 2, pp. 79–92, 1998.

[15] D. Schall, "Human interactions in mixed systems-architecture, protocols, and algorithms," Ph.D. dissertation, 2009. [Online]. Available: http://www.danielschall.at/diss/DanielSchall_PhDDefense.pdf

[16] S. Dustdar and K. Bhattacharya, "The social compute unit," *Internet Computing, IEEE*, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5755601

[17] M. Candra, "The Human-Provided Application Store," 2011. [Online]. Available: http://www.summersoc.eu/wp-content/uploads/2011/07/m.candra-summersoc-abstract.pdf

[18] S. Elnaffar and Z. Maamar, "Composite Web Services Formation Using a Social Network of Web Services: A Preliminary Investigation," *Procedia Computer Science*, vol. 5, pp. 466–471, Jan. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1877050911003851

[19] K. Osipov, "AmalgaCloud: A Technical Report," 2012.

[20] C. Hewitt, "Offices are open systems," *ACM Trans. Inf. Syst.*, vol. 4, no. 3, pp. 271–287, Jul. 1986. [Online]. Available: http://doi.acm.org/10.1145/214427.214432

[21] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 2, pp. 173–203, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1014897210525

[22] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online team formation in social networks," in *Proceedings of the 21st international conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 839–848. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187950

[23] C.-T. Li and M.-K. Shan, "Team Formation for Generalized Tasks in Expertise Social Networks," *2010 IEEE Second International Conference on Social Computing*, vol. 1, pp. 9–16, Aug. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5590958

[24] H. Yin, B. Cui, and Y. Huang, "Finding a Wise Group of Experts," pp. 381–394, 2011.

[25] M. Kargar and A. An, "TeamExp: Top-k Team Formation in Social Networks," *2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 1231–1234, Dec. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6137525

[26] T. L. Lenox, T. Payne, S. Hahn, M. Lewis, and K. Sycara, "Agent-based aiding for individual and team planning tasks," in *In Proceedings of IEA 2000/HFES 2000 Congress*, 2000.

[27] J. A. Giampapa and K. Sycara, "Team-Oriented Agent Coordination in the RETSINA Multi-Agent System," Robotics Institute, Carnegie Mellon University, Tech. Rep., 2002.

[28] M. Kargar and A. An, "Discovering top-k teams of experts with/without a leader in social networks," *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*, p. 985, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2063576.2063718

[29] M. Maghami and G. Sukthankar, "An agent-based simulation for investigating the impact of stereotypes on task-oriented group formation," in *SBP'11 Proceedings of the 4th International Conference on Social Computing, Behavioral-cultural Modeling and Prediction*, College Park, MD, USA, 2011.

[30] R. L. Rivest and C. E. Leiserson, *Introduction to Algorithms*. New York, NY, USA: McGraw-Hill, Inc., 1990.

[31] R. M. Karp, *Reducibility Among Combinatorial Problems*. Springer Berlin Heidelberg, 2010.

[32] V. V. Vazirani, *Approximation algorithms*. New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[33] H. D. Mittelmann, "Recent benchmarks of optimization software," in *22nd European Conference on Operational Research, Prague, Czech Republic*, 2007.

[34] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell system technical journal*, vol. 49, no. 1, pp. 291–307, 1970.

[35] M. E. J. Newman, "Modularity and community structure in networks." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–82, Jun. 2006.

[36] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Power in unity: forming teams in large-scale community systems," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ser. CIKM '10. New York, NY, USA: ACM, 2010, pp. 599–608. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871515