Automatic Event Scheduling in Mobile Social Network Communities

Vaskar Raychoudhury
Department of CSE, IIT Roorkee, India
vaskar@ieee.org

Ajay D. Kshemkalyani
Department of CS, Univ. of Illinois at Chicago, USA
ajay@uic.edu

Daqing Zhang
Department RST, Télécom SudPais, Evry, France
Daqing.Zhang@it-sudparis.eu

Jiannong Cao
Department of Computing, HK PolyU, Hong Kong
csjcao@comp.polyu.edu.hk

Mohit Bakshi
Department of CSE
IIT Roorkee, India
mohitbakshi2205@gmail.com

Kanik Gupta
Department of CSE
IIT Roorkee, India
kaniktopper@gmail.com

Vishal Mittal
Department of CSE
IIT Roorkee, India
coolvishal0812@gmail.com

Siddharth Maheshwari
Department of CSE
IIT Roorkee, India
siddharthm.iitr@gmail.com

Abstract: **Mobile social network (MoSoN) signifies an emerging area in the social computing research built on top of the mobile communications and wireless networking. It allows virtual community formation among like minded users to share data and to organize collaborative social activities at commonly agreed upon places and times. Such an activity scheduling in real-time is non-trivial as it requires tracing multiple users' profiles, preferences, and other spatio-temporal contexts, like location, availability, etc. Inherent conflicts among users regarding choices of places and time slots further complicates unanimous decision making. In this paper, we propose an autonomic system for activity scheduling in MoSoN communities. Our system allows flexible activity proposition while efficiently handling the user conflicts. As evident from our simulation results and analysis, our system can schedule multiple simultaneous activities in real-time while incurring low message and time cost.**

Keywords—Mobile social networks; Collaborative event scheduling; Virtual communities; Conflict resolution

I. INTRODUCTION

Mobile social network (MoSoN) [1] is a hybrid of mobile communication and social networking technologies and provides various mobility and context-aware services. In a way, MoSoN aims to improve traditional social networking experience with the use of pervasive computing [2]. MoSoN users form virtual communities [3][4] for several purposes, such as, collaborative student environment, mobile music sharing, travel and tourism [5], mobile business [6][7], etc. The virtual communities are groups of mobile users sharing the same social behaviour, interests, or goals. User communities may organize *activities* or *events* (we shall use these terms interchangeably) suited to their interests. One or more *proposers* start event scheduling at same or different times. Examples of such community activities are hiking, dining out, partying, outing, shopping, going to a movie, and many others. But due to the dynamic and loosely-coupled nature of such communities, organizing a commonly agreed upon event is non-trivial. Below we discuss more of the challenges.

Firstly, event scheduling requires collecting information regarding people's availabilities, time schedules, and preferences to join such an event. Collecting all the information manually and looking for common available time slots to schedule an activity with interested users while satisfying all user preferences is a daunting task. *Secondly*, since, there can be multiple simultaneous event proposals in a MoSoN community, a user may prefer an event proposal received later to an earlier one, and hence, may want to alter his earlier preferences. This may lead to cancellation of an event due to the want of minimum number of participants. Handling the complexity arising out of changing preferences later renders manual event scheduling more difficult. *Thirdly*, event scheduling in MoSoN commonly faces the problem of collision of interests. Preferences of users vary from each other and sometimes it becomes hard to reconcile without proper means of conflict resolution. *Finally*, manual event scheduling, if possible, is time consuming. So, rescheduling of an event as a last moment measure (due to some urgency in the part of one or more prospective participants) or scheduling an altogether different event is extremely difficult.

To address the aforementioned challenges, and to facilitate event scheduling, we propose an intelligent agent based autonomic event scheduling system for MoSoN communities. Every user maintains an intelligent *User Agent (UA)*, which resides in the portable smart device (laptop, smart phone, tab) of the user and operates with minimal user intervention. It is knowledgeable about the user's profile and preferences and keeps track of the user's real time context information, such as, his activity schedule (calendar), current location, etc. Armed with all these information, UAs of multiple users interact with each other to schedule a particular event against a proposal and inform other users about the final decision. In case, a UA is unsure about the user's preferences, it can query the user. User's input is then saved by the agent for future use. However, user preferences may vary with time and situation.

Earlier there were research carried out in the area of collaborative decision making and group consensus. As the name suggests, they focused on making decision favourable to the group members, in general, considering both small [8] and large [9][10][11] social groups. Usually, they work by developing an ideal solution based on pre-collected user's knowledge and by modelling user objective functions. Then they check existing solutions to find out the one closest to the ideal solution. However, all such works have focused on offline learning over collected

978-0-7695-5137-1/13 $26.00 © 2013 IEEE
DOI 10.1109/SocialCom.2013.63

data using AI, fuzzy logic [12][13], argumentation [14], machine learning based techniques, or optimization approaches and seem unfit to be directly applied to our problem. Our approach is more application-oriented and light-weight. Users use their mobile handheld devices to schedule an activity by collecting information and making a final decision in real time and on-the-fly. To the best of our knowledge, there is still no research focusing on event scheduling in MoSoN communities.

To summarize, in this paper, we make the following contributions.

- We propose the first (to the best of our knowledge) autonomic event scheduling system for MoSoN which works by collecting and analyzing profiles, preferences, and other spatio-temporal context information of social peers, without any user intervention.
- We propose four different parameter specification models with varied flexibility for the event proposer and they become instrumental in resolving conflicts between multiple users.
- We propose a distributed event scheduling algorithm and provide thorough analysis of time and message costs required. Our simulation results corroborate with our theoretical analysis and show that our message and time costs are significantly low.

II. Related works

Applications developed over MoSoN are finding plenty of uses in our daily lives. Location based services, such as, Foursquare [15], Jiepang [16], and iPhone BreadCrumbs [5] are allowing people to socialize while on the move. Waze [17] allows social sharing between drivers on the highways. There are many media sharing services and other related MoSoN applications to carry out plenty of tasks. In short, MoSoN connects users through sharing of location, communication, proximity, activity, status, calendar, and many other contexts.

MoSoN can also be used to build up several dedicated social communities with different goals or objectives. Recently, the SOCIETIES project [18] identifies the needs of various MoSoN communities with several pervasive computing features, like context-awareness; data and resource sharing; service provisioning; learning, reasoning and predicting; decision-making and pro-activity; security and privacy. Examples of such mobile and pervasive environments where MoSoN communities will be useful consists of scenarios [18] like, researchers in a conference, students in a university, search and rescue workers during disaster management, social car-pooling between drivers and commuters of the same route, patients and care-givers in a healthcare facility, etc. Many of the aforementioned scenarios require organizing events among community members. However, there are no such research works to facilitate autonomic event scheduling for MoSoN users.

Community detection [19][20] is an important and crucial research problem in social networks which has been extended to community detection in MoSoN [21]. This ensures the existence of community behavior in mobile social environment. However, so far little attention has been paid to schedule group activity in mobile social communities.

As already stated in Section I, during autonomic decision making processes, conflicts can often arise between several information sources (e.g., profiles and preferences of multiple users) on which the decisions are based. To the best of our knowledge, no significant research work has been carried out to resolve such conflicts. This is mainly because, projects that support adaptive behaviors, often make decisions based on a single information source. Projects such as Ubisec [22][23] and Spice [24] make decisions based on individual user's profile (specifically the user's preference set) alone and hence multi-user conflicts do not occur. We have, however, handled user conflicts in group activity scheduling and provided viable solutions for conflict resolution.

III. System Model and Event Detection Techniques

In this section we describe our data structure and system models and provide a classification of different decision making techniques.

A. Data structure

We assume that a pervasive mobile social networking environment is composed of multiple mobile users equipped with smart devices (e.g., smart mobile phones, tabs, laptops, and netbooks) connected wirelessly and they communicate through asynchronous message passing. Mobile users form a social community of like-minded users and schedule different activities and events through community-wide group decision making. Now, as already discussed in Section I, community-based decision making is a non-trivial task due to the several challenges involved. As mentioned in Section I, a proposed event is actually a group activity like, shopping, hiking, etc. We define an event (ξ) using a sextuple (U_{id}, A, T, P, N, Θ), where

- U_{id} is the identifier of the user proposing the event (ξ),
- A is the set {1 to α} of activities,
- T is the set {1 to τ} of available time slots,
- P is the set {1 to ρ} of places preferred by the user,
- N is the minimum number of users required to participate in the event (without which the event is liable to be cancelled), and
- Θ is the set {0 to 9} of integer values 0-9 and represents user preference for a particular place and/or time slot.

We assume that the duration of an activity is implicitly defined by the activity. E.g., Time to *watch a movie* is 2 hours, or *hiking* requires 6 hours, and so on. We consider that a single user can propose one or more events. Multiple users can also propose different events at the same time and all those events will be circulated across the community members in order to gather consensus in favor of them. Finally, the event with maximum number of participants (N) will be selected as a generally agreed upon one, and the information will be circulated to the prospective participants. Our proposed system models present different combinations of A, P, and T elements to account for different levels of flexibility in event proposing.

B. System Models

During launching an event, a user mainly considers the three major elements, Activity (A), Place (P), and Time (T) from the event sextuple along with their corresponding values (Θ). Considering inter-dependence of the A, P, and T elements, we propose following two different system models for the intelligent agent based event scheduling.

1) Model I

In this model (Figure 1) Place (P) is implicitly defined by activity (A), as AP. Model I (M_1) takes a single input for a combined Activity-Place (AP) and Time (T) element, where AP and T values are independent of each other. So, users can prefer (Movie-Paris, 6 PM) over (Movie-Paris, 2 PM) by entering suitable values. From Figure 1, we can observe that, due to inherent linking up of A and P, the local space in M_2 is a subset of points on the plane and is bounded by: $|A \times P| \times |T| \cong |P| \times |T| = \rho \times \tau$, where $|A| \ll |P|$.

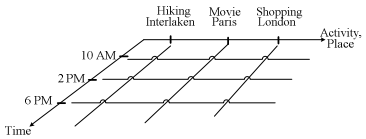

Figure 1: Model I

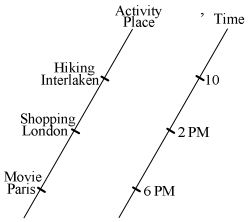

Figure 2: Model II

2) Model II

Model II (M_2) extends M_1 by further relaxing the latter. It takes two input preferences from user for two independent elements, combined Activity-Place (AP) and Time (T), and they are represented with the tuples (AP, Θ_{AP}) and (T, Θ_T), respectively. Here, user can specify fixed value for a time slot, independent of activity. E.g., a user can specify a fixed Θ_T value for some/any relaxing activity after working hours. Figure 2 shows the data points user needs to choose from while using M_2. The local space for this model is bounded by: $|A \times P| + |T| \cong |P| + |T| = \rho \times \tau$, where $|A| \ll |P|$. In M_1 and M_2, the distinction between places and activities has been removed.

IV. Our Proposed Algorithms

In this section, we present our algorithm for autonomic event scheduling in MoSoN. As stated earlier, MoSoN operates in a distributed dynamic setting, in which a user does not know the preference values of A, T, and P of other users. The objective of our algorithm is to facilitate distributed coordination among socially connected peers to schedule a commonly agreed social activity, at an agreed place and time while trying to maximize the number of participants.

A. Assumptions and Variables

When a user (U) wants to schedule an activity, he can do so by launching an event (ξ). A user can even propose an event which is not in his preference list, as long as it is popular and social. When event (ξ) is launched, it invokes our social event scheduling algorithm for the MoSoN community of U.

A mobile social network community ($\boldsymbol{C}$) can be represented as an undirected graph $G = (V, E)$, where V is the set of users in $\boldsymbol{C}$ and E is the set of social links between the users. Here, we assume that G is completely connected, so that, each user can directly communicate with every other user through asynchronous message passing. In order to save message and time costs, we propose to execute our algorithm using an overlay constructed over G.

The overlay is basically a binary tree which considers the node ids as the key values. The overlay is a dynamic structure created on-the-fly when user, U launches event ξ, and it sets U as the root. Rest of the tree nodes are chosen randomly from the set V. The overlay formation precedes the event scheduling algorithm.

We further assume that there can be multiple event scheduling going on in MoSoN community ($\boldsymbol{C}$) at the same time. E.g., if a user U_2 receives the event ξ_1 proposed by U_1 and does not feel like participating, he can initiate a new event scheduling by launching a separate event ξ_2. In order to facilitate multiple parallel event scheduling, we allow co-existence of several overlays rooted at individual nodes in V. Every root node, i.e., an event proposer maintains an array representation of their individual binary tree overlay and shares it with the nodes in their overlay in the initial stages of our algorithm.

TABLE I. Variables Used in Event Scheduling Algorithms

Variable	***Significance***
C_m	Mobile social community with m members
u	Id of user $u \in C_m$
$P_{list}[i]$	List of i places corresponding to the set P in the event (ξ) sextuple
$T_u[24]$	Time array of user u with size 24 X 1 (24-slots of 1-hour each). Each element is Boolean with values: 0 (user available) and 1 (user not available)
$LDM_u[i,j]$	Local data matrix at node u having i rows ($P_{list}.size$) and j columns ($T_u[24].size$)
P_A	Proposer of activity A, root of the binary tree overlay
$BH_A[m]$	Array representation of a binary tree overlay rooted at P_A
PAR_{uA}	Parent of node u in the overlay (for activity A), *null* for P_A
$Succ_{uA}$	Set of successors (two child nodes) of node u in the overlay (for activity A), *null* for the leaf nodes
RES_A	List of $P_i \in P_{list}$ and $T_j \in T_u$ for which consensus occurs
$FinRes_A$	Final result pair from RES_A

In Table I, we have listed the variables used for describing the pseudo-code of our proposed algorithm. LDM is a data matrix created with columns and rows corresponding to sizes of time array, $T_u[24]$ (or proposed time slots, for model I), and place list, P_{list}, respectively.

B. Algorithm Description

In this section, we shall introduce our proposed algorithm for autonomic event scheduling in MoSoN communities. We have developed one algorithm each based on model I and model II (Section III B). However, we present both the algorithms with a single combined pseudo-code in Figure 3. The codes within the red square brackets ([ ]) are used only for model I and the rest are used for model II. Since they are pretty much the same, we just consider them together as a single algorithm. However, here we shall describe them separately. Our proposed algorithm(s) operate in the following three phases.

- Phase 1: Form an overlay and disseminate ξ. Collect global knowledge regarding A, T, P of all the users.
- Phase 2: evaluate most popular activity

- Phase 3: distribute the result

We shall initially describe the algorithm based on model II followed by the algorithm based on model I.

1) Social Event Scheduling Algorithm (Model II)

We give a brief description of the algorithm for model II here.

Phase 1: A user, U, launches event ξ and communicates it through the overlay down to the leaf nodes along with the array representing the binary tree overlay structure. The leaf nodes enter their preference values for places (V_P) and send it to their parents in the tree. Thus, when the pass 1 concludes, every node knows their respective position in the overlay and the root node has the list (P_{list}) containing place preferences of all other members of his community.

Phase I @ P_A
1. Propose activity A AND create $BH_A[m]$
2. Suggest preferred place[s] P and add them to P_{list}
3. [Suggest preferred time slots T and add them to T_{list}]
4. $\forall s \in Succ_{uA}$, Send (A, P_{list}, [T_{list}], $BH_A[m]$) tuple to s
5. Wait to receive (A, P_{list}, [T_{list}]) from both child
6. Start *Phase II @ PA*

Phase I @ any non-root node u
7. receive (A, P_{list}, [T_{list}], $BH_A[m]$) from P_A; $PAR_{uA} \leftarrow P_A$
8. Find out the child nodes from $BH_A[m]$ and set to $Succ_{uA}$
9. **if** $|Succ_{uA}| \neq$ NULL,
10. $\forall s \in Succ_{uA}$, send (A, P_{list}, [T_{list}], $BH_A[m]$) to s
11. node u waits to receive (A, P_{list}, [T_{list}]) from both child and merges the P_{list}(s) [and T_{list}(s)]
12. Suggest alternate places (P); $P_{list} \leftarrow P_{list} \cup P$
13. [Suggest alternate Time Slots (T); $T_{list} \leftarrow T_{list} \cup T$]
14. Send (A, P_{list}, [T_{list}]) to PAR_{uA}

Phase II @ P_A
15. $\forall s \in Succ_{uA}$ Send (A, P_{list}, [T_{list}]) to s
16. Wait to receive (A, LDM_u) from both children
17. Invoke *LDM_processing ( )*
18. $GDM \leftarrow LDM_{P_A}$
19. Start *Phase III @ PA*

Phase II @ any non-root node u
20. receive (A, P_{list}, [T_{list}]) from P_A
21. **if** $|Succ_{uA}| \neq$ NULL,
22. $\forall s \in Succ_{uA}$, send (A, P_{list}, [T_{list}]) to s
23. node u waits to receive (A, [P_{list}, T_{list}], LDM_u) from both children
24. invoke *LDM_processing ( )*
25. send (A, [P_{list}, T_{list}], LDM_u) *to* PAR_{uA}

LDM_processing ( )
26. **if** $|Succ_{uA}| =$ NULL
27. Generate LDM_u of size ($|P_{list}| \times |T_u| [|T_{list}|]$)
/*($|\boldsymbol{P_{list}}| \times |\boldsymbol{T_u}|$) **will be replaced by** ($|\boldsymbol{P_{list}}| \times |\boldsymbol{T_{list}}|$) **for model I***/
28. **else** for both the children m and n of u
29. $\forall i \in (1, |P_{list}|)$ and $\forall j \in (1, |T_u| [|T_{list}|])$
/*(**1**, $|\boldsymbol{T_u}|$) **will be replaced by** (**1**, $|\boldsymbol{T_{list}}|$) **for model I***/
30. $LDM_u[i, j] = LDM_m[i, j] + LDM_n[i, j]$,
31. $\forall p \in P_{list}$ [AND $\forall t \in T_{list}$] provide preference rating ($R_{p,t}$) between *(0-9)*
32. $\forall t_j \in T_u$ **if** $t_j = 1$ /***not needed for model I algorithm****/
33. $\forall p_i \in P_{list}$ [AND $\forall t_j \in T_{list}$], **if** $R_{p_i[t_j]} \geq 5$
34. LDM $[i, j]$ = LDM $[i, j]$+1

Phase III @ P_A
35. MAX_{GDM} = MAX(GDM$[i, j]$)
36. Add to RES_A all (i, j) pairs for which GDM$[i, j] = MAX_{GDM}$
37. Choose one (i, j) pair from RES_A randomly and store in $FinRes_A$
38. Disseminate the final result to all the nodes in the overlay

Figure 3: Algorithm for the Model II [Model I]

Phase 2: After the Pass 1 ends, U sends the P_{list}, to the leaf nodes through the overlay. Each leaf node creates a GDM and enters their preferences (0-9) for respective places, where 0 and 9 represents the user's absolute disapproval and approval for a particular place. If the user's preference rating for the i-th place in the P_{list} is '≥' 5, then for all free time slots ($T[j]$ = 1) he increases the value of LDM $[i, j]$ by 1. After processing for each place in the P_{list} he then sends the processed LDM to his parent node in the overlay. The parent node waits to receive LDM matrix from both the children and merges them using matrix addition to generate the updated LDM. Then it adds own preference rating to that one and forwards to own parent until it reaches the root. When the root receives LDM from its children, it merges them and calls the new updated LDM as Global Data Matrix (GDM). GDM$[i, j]$ physically signifies the numbers of people who rated the corresponding place $P_{list}[i]$, greater than or equal to 5 and are available for the activity at the time $T[j]$. To find the most popular place-time combination for the proposed activity, the root node searches for the MAX(GDM$[i, j]$) which might occur for multiple i, j values. In case of multiple MAX GDM values, the proposer will have the authority to choose a particular place and a particular time.

Phase 3: This is the final pass for an activity and it involves distributing the results of the processing done in Phase 2 to the users by the proposer along with a commit deadline for the activity. The user after receiving the result has to commit to one proposer within the specified time, in case, multiple decision making algorithms are being executed in the community parallely.

2) Social Event Scheduling Algorithm (Model I)

The algorithm for model I is almost similar to model II except for few changes. Model I algorithm does not take the available time slot matrix (T_u) as input (See Table IV). Instead, the event launcher suggests some time slots and forwards it to the peer users. Other users may add other favourable time slots if they wish. Otherwise they just enter their preferences for the proposed time slots. For LDM processing while model II uses the time slot matrix, model I uses the preference rating of the user's against the proposed time slot of the event launcher.

3) Popularity-based and Preference-based Methods

We want to point out that the pseudo-code presented in Figure 3 show only the *popularity based place-time selection approach* where we consider preferences '≥ 5' as valid preferences (*Line 33*) and any preference input lower than 5 as the user's unwillingness to participate in the activity.

We shall show in the following section using examples that there is also a *preference-sum based place-time selection approach* where all input preferences for places and time slots are considered. For a particular place and time slot, preferences entered by all the users are summed up to find the highest total preference, and the corresponding place time combination is finally chosen for hosting the activity.

V. Example Applications

In this Section, we have shown two example application execution walkthrough, one for each model I and model II. Each example application considers the case of 2 parallel activity scheduling. The 2-activity case can be easily extended and applied for multi-activity scenarios. For each case, there is a popularity-based solution and a preference-sum based solution.

A. Model I: 5 Users and 2 Activities

Consider a MoSoN with 5 users with node ids 1 to 5. We assume that node 1 and 3 launch event scheduling in this MoSoN at the same or different times. The details of the proposed event are represented with the following parameters. The *broadcast tree* below gives the organization of the nodes in the binary tree overlay with the event proposer (*initiator*) as the root.

Initiator: 1 0 1 0 0 [*initiating user ids are identified with 1, rest are 0 – the initiating users have randomly been selected.*]

Name of the activity: Movie (*Activity1*) & Sightseeing (*Activity2*)

Time Length of the activity: 3 hours and 6 hours

***Name of the places proposed for Movie*:** PVR (P1), Adlabs (P2), Inox (P3), Fun Cinemas (P4), Cinemax (P5)

Name of the places proposed for Sightseeing: Delhi (P1), Agra (P2), Jaipur (P3), Roorkee (P4), Rishikesh (P5)

Time slots proposed (for Movies): 12 (T1), 15 (T2), 18 (T3)

Time slots proposed (for Sightseeing):11 (T1), 14 (T2), 17 (T3)

Broadcast tree1: 1 2 3 5 4

Broadcast tree 2: 3 4 5 1 2

User Place-time preference: *shown in the Table II (all values have been randomly generated).*

TABLE II. USER PLACE-TIME PREFERENCES

		Activity 1					Activity 2				
		P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
User1	T1	8	9	1	9	6	1	5	0	8	7
	T2	2	6	6	1	1	1	8	6	7	6
	T3	1	7	3	5	1	9	5	9	1	1
User2	T1	0	2	5	9	9	9	4	6	8	5
	T2	4	9	3	5	2	5	3	6	1	1
	T3	6	2	6	6	7	2	3	4	3	6
User3	T1	1	9	9	4	8	2	9	0	3	9
	T2	7	2	5	6	8	0	9	9	1	4
	T3	4	0	2	9	1	0	4	2	3	1
User4	T1	1	4	9	7	9	3	3	1	9	1
	T2	9	5	1	1	2	6	2	7	5	4
	T3	8	5	9	0	4	5	2	1	0	2
User5	T1	6	0	8	9	6	3	8	4	6	5
	T2	8	2	8	2	9	2	7	8	7	4
	T3	1	9	0	7	8	5	8	0	9	1

Our algorithm executes and collects various user related information to prepare the final GDM.

1) Preference Based GDM

The preference-based (or, preference-sum based) GDM is prepared by summing up respective preferences of all the users for a particular (place, time) combination. So, for activity 1 in Table II, if we add up the preference values entered by each user for the (P_1, T_1) tuple, it will give us (8+0+1+1+6) = 16. However, the addition is done from the leaf nodes towards the root following the tree structure of the overlay. Carrying out in the similar manner, the final GDM will look like the one in Table III. In Table III, the maximum value is 38 corresponding to tuples (P_4, T_1) and (P_5, T_1). The root node can choose either tuple and declare it as the final result, which says that, the users can go for a movie at 12 noon either at the Fun Cinemas or at the Cinemax. Similarly, for activity 2, the maximum value is 36 corresponding to tuple (P_3, T_2), which says that, the users can go for sightseeing at 3 PM in Jaipur.

TABLE III. GDM FOR PREFERENCE-BASED MODEL I

	Activity 1					Activity 2				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
T1	16	24	32	38	38	18	29	11	34	27
T2	30	24	23	15	22	14	29	36	21	19
T3	20	23	20	23	21	21	22	16	16	11

2) Popularity based GDM

Popularity-based GDMs consider a place's popularity before scheduling an activity at that place. This model assumes that, if a user prefers a place, he will rate it with 5 or higher score. So, if a place receives rating '≥ 5', it is considered for user's 'YES' rating about this place and 1 is added to the corresponding (place, time) tuple's current value, otherwise a '0' ('NO') is added. Calculating in this way, we can find that the final result will be that all the 5 users are ready to go for a movie at 12 noon in Cinemax, and/or for sightseeing in Jaipur at 3 PM.

Each of the preference and popularity based methods for the current example uses 40 messages to finish decision making and 13 logical time units.

B. Model II: 6 Users and 2 Activities

Consider the same mobile social network as before, only with 6 members instead of 5. The details of the proposed activity are represented with the following parameters.

Initiator: 1 0 1 0 0 0 [*initiating user ids are identified with 1, rest are 0 – the initiating users have randomly been selected.*]

Name of the activity: Movie (*Activity1*) & Sightseeing (*Activity2*)

Time Length of the activity: 3 hours and 6 hours

Name of the places (Movie): PVR (P1), Adlabs (P2), Inox (P3), Fun Cinemas (P4), Cinemax (P5)

Name of the places (Sightseeing): Delhi (P1), Agra (P2), Jaipur (P3), Roorkee (P4), Rishikesh (P5)

Broadcast tree 1 (for Movie): 1 3 6 2 4 5

Broadcast tree 2 (for Sightseeing): 3 2 4 6 1 5

Available Time slots Matrix for users (T_u*[24]*): Table IV

Place preference matrix for users: Table V

(All values in Table IV and V have been randomly generated)

TABLE IV. AVAILABLE TIME SLOTS MATRIX

User_Id	Time Slots (1-24)
U1	1 1 0 1 1 1 0 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1
U2	0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0
U3	0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1
U4	0 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0
U5	1 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1
U6	0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1 0

TABLE V. PLACE PREFERENCE MATRIX

	Activity 1					Activity 2				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
U1	8	2	1	7	1	0	6	3	1	1
U2	0	0	7	8	2	4	6	5	8	3
U3	5	2	1	3	1	4	8	9	8	4
U4	4	7	9	0	5	8	7	4	5	7
U5	0	6	7	9	5	5	0	8	6	5
U6	4	5	9	0	7	6	0	1	3	0

1) Preference Based GDM

The preference-based (or, preference-sum based) GDM is prepared by summing up respective preference values at corresponding positions in specific LDMs following the overlay tree structure. Now, the LDM of a specific node (say node *i*) is prepared by taking the cross product of a column vector (input preferences of a user for all places) and a row vector (available time slots of the same user).

$[\boldsymbol{LDM}_{u_i}] = [\boldsymbol{P}_{u_i}]^T \times [\boldsymbol{T}_{u_i}]$

⇨ $[8\ 2\ 1\ 7\ 1]^T \times [1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1]$

⇨ LDM for user 1 for activity 1 is as below in Figure 4.

	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1
8	8-8-0-8-8-8-0-8-0-0-8-0-8-8-0-8-8-8-0-8-8-8-8-8
2	2-2-0-2-2-2-0-2-0-0-2-0-2-2-0-2-2-2-0-2-2-2-2-2
1	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1
7	7-7-0-7-7-7-0-7-0-0-7-0-7-7-0-7-7-7-0-7-7-7-7-7
1	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1

Figure 4: LDM for (Preference-based) Activity 1 (Model II)

The preference-based (or, preference-sum based) GDM is prepared by summing up respective preference values at corresponding positions in specific LDMs following the overlay tree structure. E.g., the overlay tree structure for the activity 1 is given in Figure 5. The overlay tree signifies that node 2 and 4 sends their LDMs to node 3. Node 3 adds up the corresponding elements in LDM_2 and LDM_4 and with that adds own preference value to derive the LDM_3 and so on, until node 1 does the same thing. The final LDM at node 1 is called the GDM and is given in Figure 7.

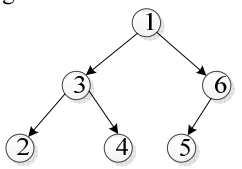

Figure 5: Overlay Tree for Activity 1

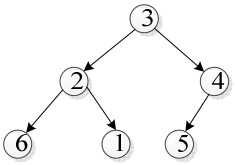

Figure 6: Overlay Tree for Activity 2

The GDM in Figure 7 shows that the highest value is 27 which is for the (P_4, T_{21}) element. This value is obtained by executing the following formula (1).

$$\sum_{i=1}^{6} U_i.P_4 \times U_i.T_{21} \ldots (1)$$

where, $U_i.P_4$ means preference of user U_i for place P_4, and $U_i.T_{21}$ means the availability of user U_i at time slot T_{21}. We have considered 6 users for this example.

↓P1 to P5 / T1 to T24 →	
08-12-09-21-16-17-05-13-04-05-17-04-08-16-04-17-12-12-13-12-17-17-17-13
08-15-15-16-20-09-02-10-07-08-15-13-02-14-05-11-13-13-14-09-15-15-15-10
08-17-24-20-26-11-08-09-16-15-18-23-01-26-09-11-17-24-19-17-25-18-18-09
16-16-20-10-16-10-11-19-08-20-19-17-07-15-00-10-16-24-03-15-**27**-19-19-19
06-11-13-14-18-09-03-07-07-08-14-12-01-15-07-07-13-15-13-08-16-14-14-07

Figure 7: GDM for (Preference-based) Activity 1 (Model II)

Replacing respective values in formula (1), gives us the following: (7*1+8*1+3*1+0*0+9*1+0*0) = 27. So, the final result says that, the users can go for a movie at 9 PM (21:00 hrs) at the Fun Cinemas. Here, we assume that all users will agree to participate and to stay for the entire duration of the movie.

2) Popularity based GDM

Popularity-based GDM creation for Model II works in the similar way as preference-based GDM (Section V.B.(1)) with a subtle difference. In this case, the user preference for a place is considered '1' only if the input preference value is '≥ 5'. Calculating in this way, we shall obtain the final results. Similarly, we can calculate the GDM for activity 2 and find the final result(s) following the overlay tree at Figure 6. The preference and popularity based methods for the current example both uses 50 messages to finish decision making and 16 logical time units.

VI. COMPLEXITY ANALYSIS

In this section we analyze the message and time complexity of our proposed algorithms. We assume that there are *n* users in a mobile social community. We further assume that the message transmission delay is negligible compared to the processing time at the nodes. We calculate the message and time complexity of our algorithms considering one phase at a time (except the last phase which is trivial). The binary tree overlay has height *h* with *i* levels, where *i* varies from 1 to *h*.

A. Phase I

1) Message Complexity

Bottom to Top

Assuming each user (leaf node) proposes equal number of places on average, Places proposed per user = *K*/*n*

So, the average size of a message packet generated at the leaf node is *c* *(*K*/*n*), where *c* is some constant.

Any internal node combines the place lists received from children nodes, and then adds its own place preferences before sending it further.

So, the average message size at an internal node at level *i*: $c*(2^{h-i+1} - 1)*(K/n)$

Summing up, total size of all messages sent from leaf to root =

$\sum_{i=1}^{h} 2^i(2^{h-i+1} - 1)*c*(K/n)$, where, 2^i = number of nodes at level *i*.

⇨ $(2^{h+1} * h - (2^{h+1} - 2))*c*(K/n)$

⇨ $(2^{h+1}(h-1) + 2)*c*(K/n)$

Therefore, the total message complexity -

$$\left[c * (2^{h+1}(h-1) + 2) * \frac{K}{n}\right] + [c * (2^{h+1} - 2) * n] = \mathbf{O}(\boldsymbol{Klog_2 n} + \boldsymbol{n^2})$$

2) Time Complexity

Top to Bottom

When the message is coming from top to bottom, every node needs to record its parent and successor nodes from the array representing the binary tree, which takes constant time, *c'*. So, the time complexity from top to bottom is *h*c'*.

Bottom to Top

Assuming T_p is the logical time unit to propose a place and it includes the constant time taken to combine places, the time taken to propose *K*/*n* places per leaf node is $(K/n)*T_p$.

Since, internal nodes combine place list received from their children, the time taken will be proportional to the size of the place list received. So, for a node at level *i*, time taken to combine the place lists is $(2^{h-i+1} - 2)*(K/n)*T_p$.

Adding the time taken to propose a place, time of processing at a node of level $i = (2^{h-i+1} - 1)*(K/n)*T_p$. Summing up time taken at all levels,

$$\sum_{i=1}^{h} (2^{h-i+1} - 1) * \frac{K}{n} * T_p$$

$$\Rightarrow \quad \frac{K}{n} * T_p * (2^{h+2} - h - 3)$$

Summing up time complexity in both top to bottom and bottom to top cases, total time complexity is:

$\frac{K}{n} * T_p * (2^{h+2} - h - 3) + h * c' = O(K * T_p + log_2 n) =$ $\mathbf{O(K + log_2 n)}$, *assuming T_p is constant.*

B. Phase 2

1) Message Complexity

Message in phase 2 primarily consists of exchanging the LDM matrix of size = *K*T* (place-time matrix).

Top to Bottom

The root node sends the place list to leaf nodes through every internal node in the overlay. So, the message size is *K*c"*.

Top-to-bottom message complexity = *n*K*c"* (one message is sent by each node, so total number of messages sent is *O*(*n*)).

Bottom to Top

Each node sends a message of size *K*T* to its parent. So, each node receives a message of size *K*T*. Total message complexity $= n * K * T + n * K * c'' =$ $\mathbf{O(n * K * T)}$.

2) Time Complexity

Top to Bottom

Since, we assume that there is no delay in message exchange, the time spent is just for processing at a node. The nodes just need to send the place list of size *K* to their children. Assuming constant time (say, *c"'*) for this job, the time complexity is *h*c"'*, where *h* is the height of the tree.

Bottom to Top

Each node rates the places. Assume time taken to rate a place is T_r, the time spent by each node in rating the places is $K*T_r$. Time to process the LDM is *K*T*c*, where, *c* is some constant.

Also each internal node combines the LDM received from both children. So, the time taken for this is *c'*K*T* where *c'* is some constant.

Total time taken at a node at level *i* is $K*T_r + K*T*c + K*T*c' = K*T_r + K*T*c''$. Summing up for all nodes,

$$\sum_{i=0}^{h} (K * T_r + K * T * c'') \; = \; h * (K * T_r + K * T * c'')$$

Total time complexity is: $h * K * (T_r + T * c'') + h * c'''$

$\Rightarrow$ $O(log_2 n * K * (T_r + T))$

$\Rightarrow$ $\mathbf{O(log_2 n * K * T)}$, *assuming T_r is constant.*

In the following section we shall present our simulation results and show that they corroborate with the complexity analysis just presented.

VII. Performance Evaluation

We have carried out extensive simulations to evaluate the performance of our algorithm. Both versions of our algorithm (model I and model II) have been simulated and compared based on proposed performance metrics.

A. Simulation Setup and Metrics

The simulation system consists of two modules: the network overlay and the social decision making algorithm. We consider a maximum of 25 users forming a MoSoN. Since, the MoSoNs are not very large conglomerations; the number of users considered can ideally model most of the usual cases.

We have simulated multiple parallel event scheduling where multiple decision making initiatives are ongoing parallely within the same MoSoN community. So, we have total 8 combinations depending on model I/model II algorithm for popularity-based/preference-based 1-activity/2-activity cases. Number of places proposed for each activity is 5. All user preference values in all experiments have been randomly generated.

We consider that the user devices in a MoSoN form a completely connected graph. As stated earlier, in order to keep a check on the number of message exchanged between users for decision making, we proposed to develop a tree-structured (binary tree) network overlay. Our algorithms are implemented as applications running on top of the network overlay. Each user device houses a software agent which accepts user input preferences (for activity, places, and time slots) and other inputs and communicates with peer agents through message exchange.

In the simulations, we measure the performance of all the algorithms using the following metrics:

- **Total decision-making time ($\mathbf{DMT_{tot}}$)** or simply, **Time** is defined as the mean time elapsed between the instant at which a node launches an event scheduling process and the instant at which it knows the identity of the nodes participating in the proposed activity. Less time to reach a decision is a measure of how efficient the algorithm is.
- **Message Overhead ($\mathbf{M_{tot}}$)** or simply, **Message** is defined as the total number of messages exchanged among all the user nodes in order to make a final decision. The less is the message overhead, the more is the saving in resource for a mobile node.

B. Simulation Results and Analysis

Below we present our simulation results with analysis. We have simulated different versions of our decision making algorithms using C++ and labeled them as *M_1_Popu_MII* (message cost in popularity based single activity model I), *T_2_Pref_MIII* (time cost in preference based dual activity model II), and similar others, like, *M_2_Popu_MII*, *M_1_Pref_MII*, *M_2_Pref_MII*, *T_1_Popu_MIII*, *T_2_Popu_MIII*, and *T_1_Pref_MIII*.

Each simulation is triggered by a single user (for '1Act') launching a new event and forwarding to neighbor nodes in the overlay. For a multi-activity scenario, any user not satisfied by the activity proposal s/he received can propose an alternate activity which will trigger a different simultaneous activity scheduling.

Each value in the simulation result is obtained by averaging over 10 different runs. We also tested for 3, 5, 6, and 30 users, but since, the results are similar, we have just omitted them. In Figures 8 and 9, we compared simulation performances of our proposed algorithms with the theoretical complexity analysis in Section VI.

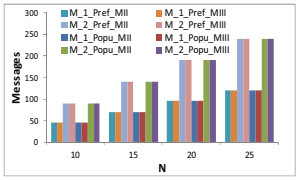

Figure 8: Message Cost vs. N

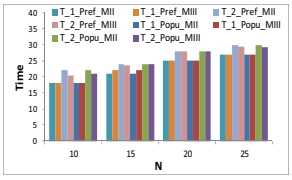

Figure 9: Time Cost vs. N

Message Complexity: The total number of messages increases linearly with the number of users, *i*.e., O(*N*), consistent with the plot results (Fig. 8). Since, the size of each message is O(*N*), the total message complexity turns out to be O(N^2) (~ ($Klog_2N+N^2$)) as calculated earlier in Section VI.

Time Complexity: In phase 1, the time complexity as calculated theoretically comes out to be O ($K+log_2N$). Now in our tests, *K* is almost constant, so the results of the testing must show a complexity of O(log_2N) which can be seen clearly in the plots (Fig. 9).

In phase 2, the logical time complexity comes out to be O ($K*T*log_2N$). So, ultimately, the nature of graph is O(log_2N), assuming $K*T$ constant, *i*.e., sub-linear which is consistent with the plot results (Fig. 9).

VIII. Conclusion and Future Works

In this paper, we proposed a novel and flexible system model and an algorithm for real-time autonomic event scheduling in MoSoN communities. We considered mainly the place and time slots which are agreed upon by all the participants and the objective was to maximize the participants in a particular social activity. Our in-depth analysis and simulation results show that our algorithm is useful and practical and incur low cost. In future, we plan to implement the algorithm in real mobile social network environments to see their usability and performance. Also, we want to further the purview of decision making by considering cross-community event scheduling which is more challenging as the interests and preferences of users change. We would like to make the system more intelligent, so that, it can proactively inform a member about other members' current activities (e.g., watching a movie together, hiking).

Acknowledgment

This research work has been partially funded by the Ministry of Human Resources and Development (MHRD), Government of India through the grant FIG-Scheme-'A' 100579-ECD and by the EU FP7 through SOCIETIES project grant.

References

[1] N. Kayastha, et al., "Applications, Architectures, and Protocol Design Issues for Mobile Social Networks: A Survey," *Proceedings of the IEEE*, vol.99, no.12, pp.2130-2158, Dec. 2011.

[2] P. Bellavista, et al., "Mobile social networking middleware: A survey," Elsevier PMC Journal, Vol. 9, Issue 4, August 2013, pp. 437–453.

[3] T. Nguyen, et al., "PlaceComm: a framework for context-aware applications in place-based virtual communities," IOS Journal of Ambient Intelligence and Smart Environments Vol. 3, Issue 1, 2011, pp. 51–64.

[4] C. El Morr and J. Kawash, "Mobile virtual communities research: a synthesis of current trends and a look at future perspectives," Inderscience International Journal of Web Based Communities Vol. 3, Issue 4, 2007, pp. 386–403.

[5] http://www.iphonebreadcrumbs.com/

[6] B.J.F. van Beijnum, et al., "Mobile virtual communities for telemedicine: research challenges and opportunities," International Journal of Computer Science and Applications, Technomathematics Research Foundation, Vol. 6, Issue 2, 2009, pp. 19–37.

[7] J. Subercaze, et al., "A service oriented framework for mobile business virtual communities," In Proceedings of *Pervasive Collaborative Networks*, Springer US, 2008, pp. 493-500.

[8] J. Fjemestad and S. Hiltz, "Experimental Studies of Group Decision Support Systems: An Assessment of Variables Studied and Methodology," In Proceedings of the 30th Hawaii International Conference on System Sciences: Information Systems Track-Collaboration Systems and Technology, 1997.

[9] M. Turoff, et al., "Social Decision Support Systems (SDSS)", In Proceedings of the 35th Hawaii International Conference on Systems Science, 2002.

[10] M.A. Rodriguez and D.J. Steinbock, "A Social Network for Societal-Scale Decision-Making Systems," North American Association for Computational Social and Organizational Science Conference, 2004.

[11] K. Nikos, et al., "Computer-Mediated Collaborative Decision-Making: Theoretical and Implementation Issues", Proceedings of the 32nd Hawaii International Conference on Systems Science, 1999.

[12] F. Herrera, et al., "A model of consensus in group decision making under linguistic assessments Fuzzy Sets and Systems," Volume 78, Issue 1, February 1996, pp. 73–87.

[13] N. Karacapilidis and C. Pappis, "Computer-supported collaborative argumentation and fuzzy similarity measures in multiple criteria decision making," Computers & Operations Research, Vol. 27, Issues 7–8, June 2000, Pages 653–671.

[14] N. Karacapilidisa and D. Papadias, "Computer supported argumentation and collaborative decision making: the HERMES system," Information Systems, Vol. 26, Issue 4, June 2001, pp. 259–277.

[15] https://foursquare.com/

[16] http://hk.jiepang.com/

[17] http://www.waze.com/

[18] http://www.ict-societies.eu/files/2011/11/SOCIETIES_D22.pdf

[19] S. Fortunato, "Community detection in graphs," In Physics Reports, Vol. 486, No. 3, 2010, pp. 75-174.

[20] M. Plantié and M. Crampes, "Survey on Social Community Detection," In Social Media Retrieval, Springer London, 2013, pp. 65-85.

[21] P. Hui, et al., "Bubble rap: social-based forwarding in delay tolerant networks," In Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), 2008, pp. 241-250.

[22] Ubisec STREP Project, 6th Framework Programme, Homepage URL: http://www.clab.de/en/researchprojects/completed-research-projects/ubisec/index.html, January 2004 – February 2006.

[23] J. Groppe, and W. Mueller, "Profile Management Technology for Smart Customizations in Private Home Applications," In Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA), 2005, pp. 226-230.

[24] C. Cordier, et al., "Addressing the Challenges of Beyond 3G Service Delivery: the SPICE Service Platform," Workshop on Applications and Services in Wireless Networks (ASWN '06), 2006, Berlin.