Cluestr: Mobile Social Networking for Enhanced Group Communication

Reto Grob
Swisscom Strategy & Innovations
Switzerland
reto.grob@swisscom.com

Michael Kuhn
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
kuhnmi@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

Martin Wirz
Wearable Computing Laboratory
ETH Zurich, Switzerland
martin.wirz@ife.ee.ethz.ch

ABSTRACT

Recent technological advances foster the spreading of social software in the mobile domain. Hence, future usage patterns of mobile devices will involve more group interaction. While collaboration using mobile devices is an active area of research, only limited attention has been paid to the efficient initiation of group communication from mobile terminals. In this paper we present a community-aware mechanism that allows to efficiently select contacts in order to address them as a group. We have integrated the proposed method into a proof-of-concept application, and present preliminary experiments that demonstrate the accuracy of the approach and show significant time savings in the group initialization process.

Categories and Subject Descriptors

H.5.3 [**Information Systems**]: INFORMATION INTERFACES AND PRESENTATION—*Collaborative computing, Group and Organization Interfaces*

General Terms

Human Factors, Algorithms

1. INTRODUCTION

Over the past years, social services have experienced a tremendous success. Sites such as MySpace, Facebook, or LinkedIn have steadily been growing and became ubiquitous on the Internet. Nowadays, many people can no longer imagine a life without such applications.

Interestingly, at this stage, the success of similar services in the mobile domain is significantly smaller. Different reasons are said to be responsible for the missing success. Restricted input and output capabilities of mobile devices have inhibited the spreading of a "Mobile Web 2.0". Moreover, inappropriate communication technologies that offer low data rates at high prices are often seen as an important reason for the low acceptance of the corresponding services. Recent developments, however, mitigate these issues. The mobile infrastructure is migrating to a packet switched all-IP network. Wireless broadband technology is constantly improving, and 3G coverage is spreading rapidly. In addition, in many countries, a shift from time to data centric and even flat rate pricing models can be observed, which allows for permanent connectivity at low cost. Finally, state-of-the-art mobile phones, such as the iPhone and Android based devices, have literally revolutionized the user interface side. Moreover, rich APIs and transparent application distribution channels are major drivers for innovation. Considering these technological advances, it seems that the potential of mobile devices can finally be fully exploited. Thus, we expect that more and more social services will expand their reach into the mobile domain.

Forerunners of this trend are specific mobile versions of services such as MySpace[1], LinkedIn[2], and Facebook[3] that have recently experienced an increasing popularity. However, currently available services do not exploit the full potential of the underlying technology. In particular, they often merely try to copy successful concepts from their desktop counterparts. However, the usage patterns of mobile phones significantly differ from those in a wired environment. It is thus crucial that mobile services take the special characteristics of the environment, such as mobility, ubiquity, or permanent reachability into account.

The wired telephony network once lowered the communication delay and made spatial distance between communicating parties negligible. Today's mobile infrastructure continues this trend: We can reach everyone everywhere all the time – instantly. As a result, the differences between personal and remote communication patterns are diminishing. Thus, the 1:1 conversation scheme known from traditional remote communication will more and more be complemented with group interaction, much as when socializing in the real world. Future-oriented mobile social services thus have to adequately address the related issues. In particular,

GROUP'09, May 10–13, 2009, Sanibel Island, Florida, USA.

[1] m.myspace.com
[2] m.linkedin.com
[3] m.facebook.com

they should facilitate group communication and offer support for collaboration. As we will see in the next section, several ongoing research activities are devoted to this field.

One important aspect falls short in these activities: The initial group formation process. A major focus of this paper thus lies on the fast and convenient initialization of group interaction on mobile devices. In a survey we identify relevant requirements to enhance group communication on mobile devices. Based on the findings, we then propose a contact recommendation algorithm which allows to quickly group people in order to contact them at once. Moreover, we outline mechanisms that simplify the interaction within such a group after its initial formation. The most important components have been implemented in Cluestr, a proof-of-concept application running on Windows Mobile devices. Based on this application, we have conducted a preliminary user experiment. The experiment makes use of real-world data extracted from Facebook to demonstrate the suitability of the proposed methods. In particular, we demonstrate that relevant community information can accurately be deduced from this data. Moreover, we show that, when using our recommendation engine as opposed to traditional list-based approaches, in realistic scenarios significant time savings can be achieved during group establishment.

2. RELATED WORK

Traditional online social networking platforms provide only marginal functionality for enhanced communication on mobile devices. By contrast, truly mobile social services take people's mobile behavior patterns into account and offer functionality to enhance communication. Purely decentralized friendship exploration has been addressed by Veneta [24]. Services like TXTMob [11], Jaiku[4] or Twitter[5], on the other hand, implement a broadcasting system to which friends can subscribe in order to receive message updates. FriendFeed[6] enables friends to comment on such messages and thereby extends this approach. ContextContact [20] and Swarm [5] are designed to enhace communication within a large group including all of a user's contacts.

Communication in mobile space is often used to plan, schedule, and reflect on group activities. As stated in [12], mobile phones are not ment to support such behavior in a group context. The authors present the idea of creating privately shared group spaces on mobile devices where each group is able to communicate and collaborate in order to overcome this lack of functionality.

Similar ideas have been followed in Microsoft's SLAM project [3] and in PlaceMail [14]. In these projects, the focus lies on communication within small groups e.g. class mates, co-workers or family members.

Most mobile social applications provide some degree of support for collecting contextual information and posess functionality to publish or share this information with other members of a group. There exist plenty of applications that offer such context sharing: Some systems disclose information about their users' presence [20, 23], others about location [19, 22], motion [1] and proximity [21].

Besides sharing contextual information among members of groups, some services incorporate collaborative functionality. Besides group-based communication, some services provide functionality to collaboratively solve tasks using mobile devices. An example is Doodle[7], which offers a polling service for multi-party negotiation.

All approaches have in common that the group formation has to be done manually. The systems are not able to automatically deduce community affiliation from user behavior.

We next present a survey which indicates that current group establishment mechanisms do not well agree with the users needs and that more efficient methods are demanded.

3. SURVEY

The previously mentioned technological and economical developments ask for new concepts and ideas regarding mobile communication. To shed light on the usage patterns of today's mobile phones, we set up an online survey. It focuses on aspects related to group communication and collaboration in mobile settings and investigates to what extend social networking services could contribute in this context. A total of 342 people from Europe participated in the survey. The outcome can be summarized as follows:

- Most participants agreed on the fact that their contacts stored in the mobile phone's address book can be grouped into communities such as 'university colleagues', 'coworkers', 'family', 'friends' etc., and that communication often occurs among the members of a certain community simultaneously. However, although today's mobile phones allow to group contacts, this functionality is hardly used. Only 16% use the built in grouping function to assign contacts to groups.

- 68% of all participants use the feature to send text messages (SMS) to multiple receivers. It is being used for different tasks including holiday greetings, invitations, scheduling meetings, event organization and polls.

- The conference call feature is not often used in daily life. 11% claim to use it on a non-regular basis. Only one person claimed to use it regularly. The main purpose for using this feature are business meetings.

In particular, we conclude that:

- The mobile phone is often used to organize and coordinate activities among multiple people, such as to discuss how to spend the evening, or to decide about a meeting point and time.

- Group communication is often performed with members of a community existing in real life, such as members of a sport team, coworkers, class mates or family.

- Existing features for group maintenance and group communication are rarely used. Rather, most people manually (re-)establish groups, when they, for example, want to send a message to multiple receivers.

Today's mobile communication alternatives do not cope with this social behavior. Text messages can be sent to multiple receivers simultaneously. However, the underlying system is limited with respect to group communication. In

[4]www.jaiku.com
[5]www.twitter.com
[6]www.friendfeed.com
[7]m.doodle.com

particular, text messaging only offers a 1:N way of communication. For efficient group communication, however, an N:N solution is required.

The conference call feature included in the majority of state-of-the-art mobile phones offers such functionality. However, it is restricted to voice communication and only works in synchronous mode, i.e., all the participants have to concurrently be present (at the phone) in order to receive the information. E-mail does not suffer from the outlined problems. However, its popularity in the mobile context is still low in many regions. Reasons are, presumably, a insufficient integration in current devices, together with the limited input capabilities of these devices.

Finally, only a minority of users uses the built in feature to maintain groups of contacts. A major reason for the low acceptance is presumably the high dynamics involved. People join and leave communities (e.g. a sport team), which implies tedious work to keep the group information up-to-date. A main contribution of this paper is an interface that simplifies group formation without the need for permanent maintenance.

4. CLUESTR

In the following, we outline a service which focuses on group communication and collaboration, and addresses the major findings of the survey. The service is named Cluestr, a combination of *cluster* and *clue*. Cluestr should be seen as a proof-of-concept implementation and an evaluation environment. It is not our goal to promote yet another mobile social application. The main ideas and concepts addressed in this paper can easily been integrated into existing systems.

4.1 Vision

Cluestr is designed to enhance communication among members of a group. In particular it helps to lower the effort to organize, manage and coordinate group activities, such as visiting a cinema, which requires a group of friends to agree on a movie, meeting point and time. A further enhancement is the incorporation of collaboration capabilities that can be used among members of a group. The following illustrative example sketches a typical situation where Cluestr is useful:

> Every Saturday, a local football team has a match against a rivaling team. Using Cluestr, the team captain can initiate a group and invite all team mates as participants. On a billboard, team mates can then inform the others whether they will join the game or not and discuss about the meeting point. Using a poll function, they can vote for the person who has to be the chauffeur and drive to the game. Using a ToDo list, the team manages the logistics for the BBQ afterwards. Everyone can tick what he will contribute to the buffet.

The strength of the Cluestr service lies in its support for group communication and collaboration, and in particular in a novel approach to initiate groups. The participants of such groups can profit from intragroup communication and collaboration tools, such as a thread-like billboard, where everyone can post and read messages. In addition, a poll, where participants can vote, and a ToDo list where participants can add elements or mark them as accomplished is available for each individual group. In the following we focus on the group initialization process.

4.2 Need for an Efficient Group Initialization

The main contribution of this paper is an efficient method to establish a group. Groups are initiated by one person (initiator), who is then able to invite a set of contacts to participate.

Due to interface constraints, inviting contacts should be kept as simple and intuitive as possible. Selecting participants from a traditional, alphabetically ordered contact list is inefficient since the required contacts are in general randomly distributed over the whole range. Cluestr offers a more efficient method for finding desired contacts. We present a contact recommendation engine able to support the initiator by suggesting suited contacts for invitation. For illustration, we give a simple use case:

> If the initiator of a new group invites contact A for participating, the engine proposes that contacts B, D and E might also fit into this group but not contact C. The initiator decides to invite E as well. With this additional information, the engine understands that the initiator is, say, not interested in D and hence only recommends B.

In addition to time-saving, contact recommendation helps to remind the initiator of contacts that he/she otherwise might have forgotten, or that he/she was even not aware of.

4.3 Concept of the Contact Recommendation

The principles behind our recommendation algorithm are based on the outcome of our survey: The initiator belongs to different communities and often communicates with several members of a community simultaneously. The basic idea is to take advantage of existing community structures in the following way: Whenever a user wants to initiate some form of group interaction, he/she initiates a new group and starts by selecting a first contact. Next, the engine proposes a list of people that share one or several communities with the selected contact. This list is sorted by relevance, which is given by the number of shared communities. The user can now select a next contact from this list, which in turn gets repopulated with the entries best matching both selected contacts. This process is repeated until the group is complete.

One could think of different approaches to extract the required community affiliations of the initiator's contacts:

- Tagging of contacts and manual grouping
- Semantic analysis of the communication content
- Communication pattern analysis
- Social graph topology

Tagging and manual grouping of contacts implies a large maintenance effort by a user, which is undesired. Moreover, this feature is already implemented in many mobile phones but hardly used according to our survey. We want our recommendation engine to be able to recommend relevant contacts with the least possible user effort.

Analyzing the content of the communication to then group contacts around topics might be another approach. This

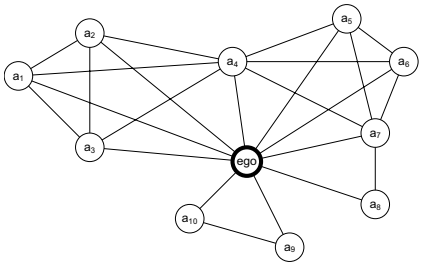


Figure 1: Visualization of a sample ego-graph

idea is followed in [4, 6, 10]. However, a lot of mobile communication is voice based. Gaining relevant content information using voice recognition is hard to achieve on mobile devices.

One could also imagine that community affiliations are deducible by observing the stream of communication on the user's device. First experiments we have conducted in this direction, however, indicate that solely analyzing communication patterns does not allow to estimate community structure with sufficient accuracy. It remains open, though, whether the required accuracy could be reached if further contextual cues were included.

The approach followed in this paper is based on social network analysis. In a social networking service, users link to each other to indicate relationships. This leads to a network in which related users are connected through ties. We designed our recommendation engine to require only this network information to find community affiliations of contacts and use this information to generate recommendations.

5. CONTACT RECOMMENDATION

The previously introduced concept of contact recommendation requires the engine to know:

- The different communities the initiator belongs to.
- Which of the initiator's contacts belongs to which communities.

As mentioned before, this information is extracted from the social network stored on the Cluestr server.[8] Before describing the extraction process in detail, we introduce some notations.

5.1 Notations

Under a *social network* (or *social graph*) we understand a graph $G = G(V, E)$, where the set V of vertices represents users (or contacts), and the set E of edges denotes links (or friendships) between these users. An *ego-graph* (or *egocentric graph*) is a special form of a social graph. It consists of a user of interest (*ego*) and his/her direct neighbors (*alters* a_i).[9] A sample ego-graph is illustrated in Figure 1.

A *cluster* is a subset of vertices of a social graph that is highly connected. In particular, we assume that the density of edges within a cluster (intra-cluster edges) is larger than the density of edges connecting vertices from inside the cluster to vertices outside of the cluster. A *clustering algorithm* seeks to partition a graph into a set C of clusters c_i. The resulting partition is also called the *clustering* of a graph. If the resulting clusters do not need to be disjoint, the clustering is said to consist of *overlapping clusters* (i.e. one vertex can belong to several clusters).

[8]Observe that all the presented concepts could also be based on an existing social networking service (such as Facebook).
[9]The notation (ego, alter) is taken from [25].

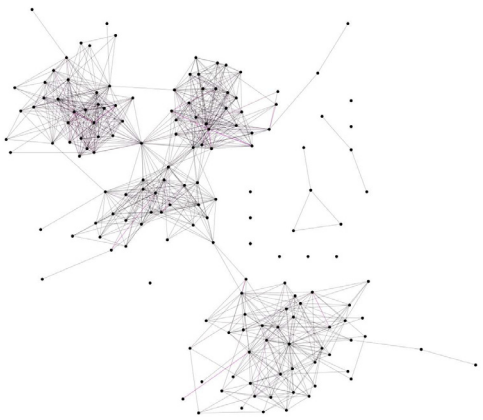


Figure 2: A real ego-graph: The characteristic community structure is clearly visible. The central user (ego) as well as the ego-alter ties are not displayed for simplicity.

As we will see, clusters are supposed to reflect real community structure. Throughout this paper, we will refer to a *cluster* c as a group of contacts, as identified by our clustering algorithm. By contrast, a *community* o refers to a group of contacts as identified by a user. Similarly as we denote the set of all clusters by C, we refer to the set of all communities by O.

Modularity is a well known measure to estimate the quality of a clustering. It was introduced by Newman [16]. Basically, the modularity is defined to be the fraction of edges that fall within the given clusters minus the expected fraction of such edges, if edges were distributed at random. The original definition of modularity is only valid for disjoint clusters. However, Nicosia et al. [18] have recently proposed a variant which is also usable with overlapping clusters. We refer to this second definition when using the term modularity.

5.2 Community Detection

Figure 2 shows the topology of a real ego-graph retrieved from Facebook. The example graph exhibits a characteristic structure, which is common to most real-world ego-graphs. There are groups of alters that are densely connected, but only sparsely interlinked to the rest of the network. These dense regions exist due to the social characteristics of communities. The members of a community typically know each other, and thus form densely connected subgraphs. On the other side, members of one community do often not know members of other communities, resulting in a sparse interlinkage.

As mentioned before, densely connected regions can be extracted from a graph using a clustering algorithm. Graph clustering is an active area of research and a wide range of algorithms have been proposed, including [2, 7, 15, 16,

17]. Due to the outlined correlation between interlinkage and communities, we expect that the clustering generated by a sophisticated algorithm well reflects the actual community structure. A careful look at Figure 2 reveals that some of the alters belong to more than one community. A clustering algorithm which is supposed to extract the actual communities thus needs to be able to deal with overlapping clusters. An algorithm that fulfills this property is the CONGA algorithm [8]. It extends the widely used Girvan and Newman algorithm [7, 17], which can only retrieve disjoint clusters.

CONGA belongs to the class of divisive hierarchical clustering algorithms. That is, it starts with a single cluster containing all vertices and, by subsequent partitioning, produces an ever finer grained hierarchy of clusters. The partition process can be repeated until all nodes form single node clusters, which are the leaves of this hierarchy. To explain the algorithm in more detail, we use the notion of *edge betweenness* and *split betweenness*, according to [8]:

- *Edge betweenness*: The betweenness of edge e is defined as the number of shortest paths, between all pairs of vertices, that pass along e. A high betweenness means that the edge acts as a bottleneck between a large number of vertex pairs and suggests that it is connecting different clusters.
- *Split betweenness*: A vertex v can be *split* by partitioning the set of its neighbors into two disjoint sets s_1 and s_2. Then, v is replaced by two virtual nodes v_1 and v_2 that are connected by an edge e_v. Moreover, each node in the set s_i is connected to v_i. The *split betweenness* is defined as the maximum edge betweenness of e_v among all possible partitions into s_1 and s_2. Since the number of possible partitions ($2^{\delta-1}-1$, where δ is the degree of node v) can be high, an approximation algorithm to calculate split betweenness is used by [8].

The CONGA algorithm can then be described as follows:

1. Calculate edge betweenness of edges and split betweenness of nodes.
2. Remove the edge with maximum edge betweenness or split the node with maximum split betweenness, if higher.
3. Recalculate the edge and split betweenness values.
4. Repeat from Step 2 until no edges remain.

In [8], the time complexity of the CONGA algorithm is shown to be $O(m^3)$, where m is the number of edges in the graph. Gregory has later proposed a modified version of the algorithm (named CONGO) to reduce the time complexity [9]. However, we found that performance is not a key issue, and, as it provides more accurate results, rely on the original CONGA algorithm for ego-graph clustering.

5.3 Recommendation

As discussed before, we assume that most group interaction takes place within communities. Therefore, the chance that the initiator of a group wants to invite several members of the same community is high. The automatic detection of dense structures (clusters) in the initiator's ego-graph approximates the real-life communities like class mates, work colleges, family members, and so on.

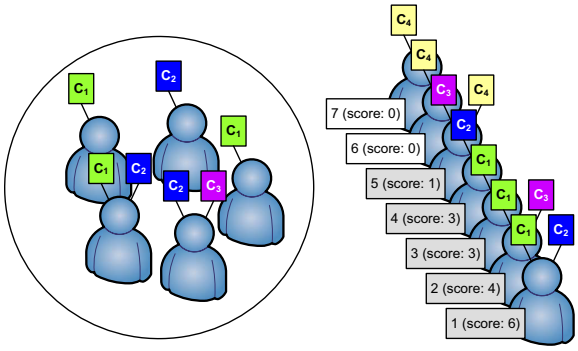


Figure 3: Contacts are rated according to their cluster affiliations. For each occurrence of a cluster within the already invited group members (left), a contact's score increases by 1. For the top ranked user (that belongs to clusters 1 and 2), for example, this amounts to a total score of 6.

The recommendation process for the initiation of a new group thus works as follows:

1. Detect hidden community structures by clustering the initiator's ego-graph. Each contact is assigned to at least one cluster.
2. Present an alphabetically sorted list from which the initiator can choose a first contact.
3. Rank contacts based on the cluster affiliations of the previously selected contacts.
4. Recommend the best ranked contacts to the initiator.
5. Continue with Step 3 after the initiator has selected a next contact to invite, or terminate if the group is complete.

The idea of the ranking function in Step 3 is to rank contacts high that share many clusters with the already invited persons. In particular, each cluster is weighted with the number of occurrences within the already selected contacts. Each of the remaining contacts is then scored by the sum of all the clusters it resides in (see Figure 3). If the list from Step 4 does not contain any relevant contacts, the user can switch back to an alphabetical list to continue the invitation process. Often, the recommendation algorithm recovers in the next iteration, such that an alphabetical list has to be consulted only rarely. Assume, for example, the user wants to invite both, team mates as well as co-workers, to a birthday party. If these communities – and the corresponding clusters – are entirely disjoint, the recommendation process will work fine for the first cluster, then require the consultation of the alphabetically ordered list, and afterwards work fine again for the second cluster.

6. EVALUATION

We have investigated different aspects of the proposed contact recommendation algorithm, namely:

- *Clustering accuracy*: How well do the clusters generated by the clustering algorithm reflect the social communities identified by the user.

- *Advantage of recommendation*: How much time can be saved if a group is created by using our recommendation algorithm rather than by selecting contacts from an alphabetical list.
- *Effect of sparsity*: How well does the proposed concept work if data is spare, e.g. due to a low number of registered users, or due to missing friendship information in contact books.

Evaluating an algorithm on real social graphs is challenging. It is impossible to characterize the performance of the algorithm without knowledge of the correct community structure, the *ground truth*. For proper analysis, therefore, not only information about the social graph but also about existing communities is required. Such information can best be retrieved by subject questioning.

Extracting the ego-graph from real mobile phone contact books is infeasible.[10] We thus decided to rely on Facebook data. Facebook is the presumably most complete and well established online social networking service. The underlying social network is thus likely to exhibit similar characteristics as the social network defined by the contact information in people's mobile phone address books.

Due to the lacking ground truth, available data sets as presented in [13] cannot be used for evaluation. For our experiments, we thus extracted the ego-graphs of four subjects (two male, two female) from Facebook. Moreover, we asked these subjects to group their (Facebook) friends into an arbitrary number of communities (such as coworkers, team mates, family members, etc.). The resulting four datasets form the ground truth for our analysis. Each of the datasets contains between 59 and 151 contacts, and was assigned between 4 and 7 communities by the corresponding subject.

6.1 Evaluation Measures

In the context of classification tasks, *precision* and *recall* is a widely used evaluation concept. The *precision* for a class denotes the number of items correctly labeled as belonging to the class (i.e. true positives) divided by the total number of elements labeled as belonging to the class (i.e. the sum of true positives and false positives). Similarly, *recall* is given by dividing the number of true positives by the total number of items that, according to the ground truth, really belong to the class (i.e. the sum of true positives and false negatives).

In our setting, we are interested in the accuracy of the extracted *clusters* with respect to the *communities* identified by our subjects. Above measures thus become to:

- Precision of cluster i with respect to community j:

$$P_{i,j} = \frac{|\{\text{contacts in } c_i\} \cap \{\text{contacts in } o_j\}|}{|\{\text{contacts in } c_i\}|}$$

- Recall of cluster i with respect to community j:

$$R_{i,j} = \frac{|\{\text{contacts in } c_i\} \cap \{\text{contacts in } o_j\}|}{|\{\text{contacts in } o_j\}|}$$

These two values are often combined to form a single evaluation measure, called *F-measure*, which is the harmonic mean of recall and precision:

$$F_{i,j} = 2 \cdot \frac{P_{i,j} \cdot R_{i,j}}{P_{i,j} + R_{i,j}}$$

[10]Extracting the entire ego-graph would require to read the contact books of all contacts of ego.

Observe that the assignment of a cluster c_i to a community o_j is not known beforehand. We thus chose the o_j that maximizes the *F-measure* $F_{i,j}$ as the representative community $o_{j^*(i)}$ of c_i, i.e., $j^*(i) = \text{argmax}_j F_{i,j}$. The corresponding measures for the entire classification task can then be defined as follows:

- *Precision:*

$$P = \frac{1}{|C|} \sum_{i=1}^{|C|} P_{i,j^*(i)}$$

- *Recall:*

$$R = \frac{1}{|C|} \sum_{i=1}^{|C|} R_{i,j^*(i)}$$

- *F-Measure:*

$$F = \frac{1}{|C|} \sum_{i=1}^{|C|} F_{i,j^*(i)}$$

6.2 Clustering Accuracy

The contact recommendation algorithm can only work if the ego-graph decomposition of the clustering algorithm well agrees with the intuition of the user. The CONGA algorithm has shown to perform well in various settings [8, 9]. Nevertheless it remains to be shown that the graph structure and the resulting clustering well enough represent the perception of a user about how to naturally group his/her contacts.

We therefore investigate the accuracy and applicability of CONGA for the partition of an ego-graph into communities. First, a suitable termination criteria for the divisive clustering process is required, which leads to a well defined number of – possibly overlapping – clusters. As pointed out by Gregory [9], modularity can, but does not need to be an appropriate criteria for this purpose. Our first experiment thus investigates whether modularity maximization also maximizes the F-measure. We clustered each ego-graph until singleton clusters remained. At each *stage*, i.e. for each number of clusters, recall, precision and the resulting F-measure were calculated. A plot of clustering stages 4 to 50 is given in Figure 4.[11] The vertical line indicates the clustering stage at which modularity is maximized. In all but one case (Subject 4), the clustering stage that optimizes modularity agrees with the stage that maximizes the F-measure. Moreover, for Subject 4, it differs by only one cluster. This good alignment indicates that modularity is a well-suited optimization criteria for clustering ego-graphs.

While we have seen that modularity is a good criteria to maximize the F-measure, it is still not clear whether the resulting clusters reflect the user's perception well. In particular, the number of clusters detected by the algorithm (at maximum modularity) might greatly differ from the amount of communities identified by a subject. Table 1 compares the number of manually defined communities to the number of clusters that maximize modularity and F-measure, respectively.

The fact that these numbers all lie in the same range indicates that the modularity maximizing CONGA clustering well reflects the users' intuition concerning communities. However, we did not reach perfect correspondence.

[11]The evaluation starts as stage 4, since the used algorithm implementation did not allow to create clusterings with fewer than 4 clusters.

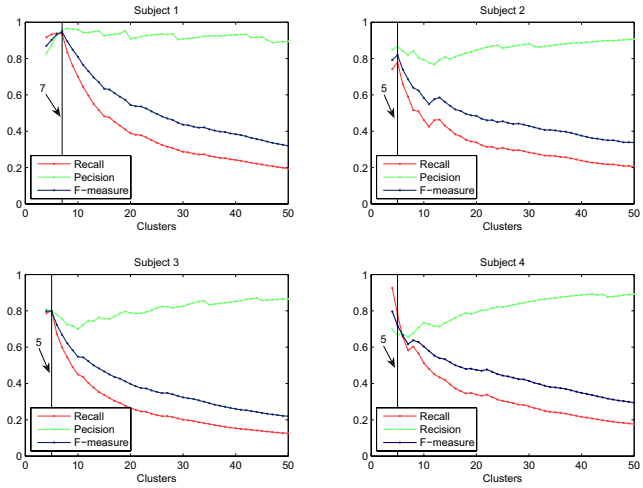


Figure 4: Comparison of the accuracy of clustering the subject's ego graph at different stages. The vertical line indicates the stage with maximal modularity.

	#Communities	#Clusters (max Mod)	#Clusters (max F-measure)
Subject 1	7	7	7
Subject 2	7	5	5
Subject 3	4	5	5
Subject 4	7	5	4

Table 1: Comparison of the number of clusters resulting from subject questioning and CONGA.

To understand the reasons for the differences, we asked the subjects to name their communities and also give names to clusters recognized by the algorithm. A comparison of the outcome facilitates a qualitative interpretation. Basically two effects caused the number of clusters to differ:

- Combining two independent groups of friends, which are barely interlinked, into one community. Such communities are detected as different clusters by the algorithm. One subject, for example, put all friends she got to know during her internship into one community. However, this community was recognized as two clusters containing flat mates she was living with in one cluster and the other containing co-workers. There were no ties connecting these two groups.

- Two communities were discovered as only one when the interlinkage between friends was too high to separate them. This may happen when one community is a subset of another one. One subject, for example, put her friends from university into one group and defined a second community with friends she knows from a student organization. Members of this organization, however, go to the same university and possess friendships with other students not in the organization. This made it impossible to separate the two communities. As a consequence only one cluster was detected.

To quantify the accuracy of the clustering, we have applied the aforementioned quality measures, namely precision, recall and F-measure. Table 2 summarizes the results. The high F-measure values (average: 82%) achieved by automated clustering are promising especially when taking the two above stated causes for non-congruency into account. We conclude that a user's ego-graph contains sufficient information to realize a contact recommendation system, and that this information can be efficiently extracted by the CONGA algorithm.

Subject	Recall	Precision	F-measure
Subject 1	0.94	0.96	0.95
Subject 2	0.78	0.87	0.82
Subject 3	0.80	0.79	0.80
Subject 4	0.78	0.67	0.72
Average	0.83	0.82	0.82

Table 2: Clustering accuracy (recall, precision and F-measure) for each subject.

6.3 Recommendation Engine

In the previous section, we have shown that clustering an ego-graph using CONGA at maximized modularity results in an adequate estimation of communities and their affiliated contacts. Next, we evaluate whether recommendation based on graph clustering is able to improve the group initialization process. We want to analyze the performance of our recommendation engine. For this, we asked the subjects to establish groups and send invitations to friends from their contact list using Cluestr. Three different groups had to be established according to different scenarios:

1. "Invite all members of a community of your choice for a BBQ."

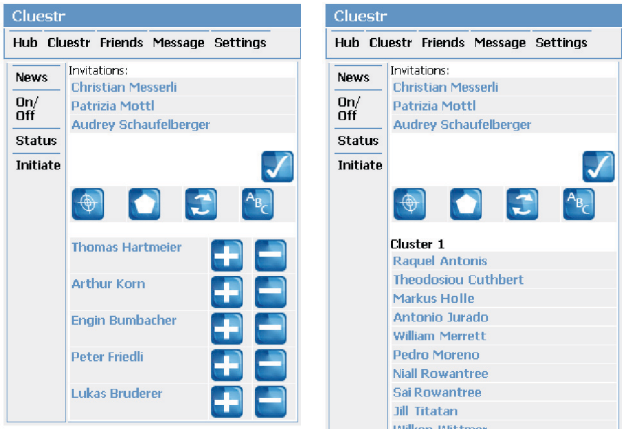
2. "Invite some of your contacts from one community of your choice to watch a movie at your home."

3. "Invite a random selection of contacts for participation in a user study."

The scenarios pose increasing challenges to the recommendation engine. In the 1. Scenario, the group consists of all members of one community. In the 2. Scenario, only a part of a community's members should be invited. The 3. Scenario, finally, deals with a random sample of contacts, regardless of their community affiliation.

The experiment was performed on a mobile device[12] using Cluestr. The subjects' ego-graphs from Facebook were used as the data set for this experiment. Each task had to be solved following the same procedure: In a first step, the subjects were asked to write down all participants they wanted to invite. Then, these participants had to be invited using Cluestr. Each subject had to perform the invitation procedure on the device three times. First, the subject was shown a traditional *alphabetic list* of his/her friends, second, he/she had to establish the group using a *cluster view*, where friends are grouped according to the detected clusters (see Figure 5(a)). Third, the subject had to choose the participants using the *recommendation view*, in which the subjects could select contacts from the top 5 recommendations (see Figure 5(b)). During the experiment, the time required to initiate a group was measured. Figure 6(a) shows the average time required to select a participant in each scenario

[12]HTC Touch Cruiser



(a) Group initialization process: recommendation view (b) Group initialization process: clustering view

Figure 5: Screenshots of Cluestr running on a Windows Mobile device

and each selection mode. Figure 6(b) plots the mean values over all subjects.

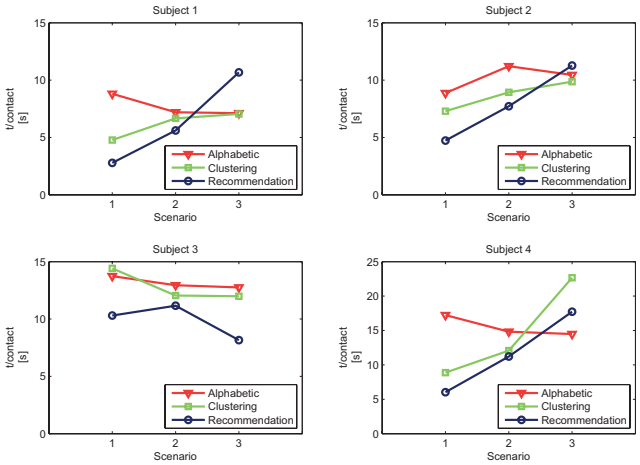
The results show that the recommendation algorithm performs stronger, the more community-centric the group is. The more randomized the selected contacts are (in terms of community affiliation), the more challenging it gets for the recommendation engine to come up with adequate suggestions. In a completely randomized situation, as evaluated in Scenario 3, recommendation performance thus decays.

During our experiment, whenever the recommendation was inappropriate, the subject could switch to either the alphabetic or clustered list to select a next participant. The less community focused the group was, the more frequent a subject had to switch to these selection modes. However, our survey indicates that most groups are established with contacts that form a community in real life. Therefore, Scenario 1 and Scenario 2 are more realistic.

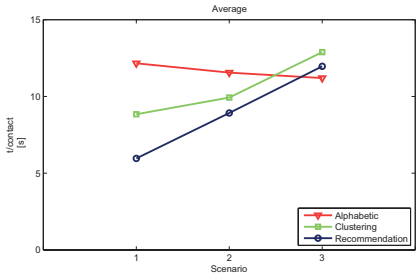
These results show that both, listing contacts according to cluster affiliation as well as recommendation-based selection, results in time saving. Contacts can be found faster and in a more convenient way. In our experiment, the average selection time per user in Scenario 1 could be cut in half from 12.2s to 6s by using our recommendation engine. If only a partial, rather than an entire community needs to be selected, a reduction of 23%, from 11.5s to 8.9s, was achieved. The average size of the initated groups consisted of 19 contacts, resulting (in average) in an overall timesaving of 120s and 50s, respectively. Besides the improvement in time, the subjects also mentioned that the recommendation helps to remember all relevant participants and thus reduces the risk that somebody is unintentionally not invited.

6.4 Clustering Stability

As mentioned before, we assume that the ego-graphs retrieved from Facebook are well established and therefore stable. Typically, people who are registered on Facebook are already connected with most of the relevant users. Moreover, for a single user, a significant amount of friends are likely to be registered on Facebook. If a new social networking service is launched, the situation is quite different.



(a) Time required to initiate a group by each subject for each scenario.



(b) Average time required to initiate a group for each scenario.

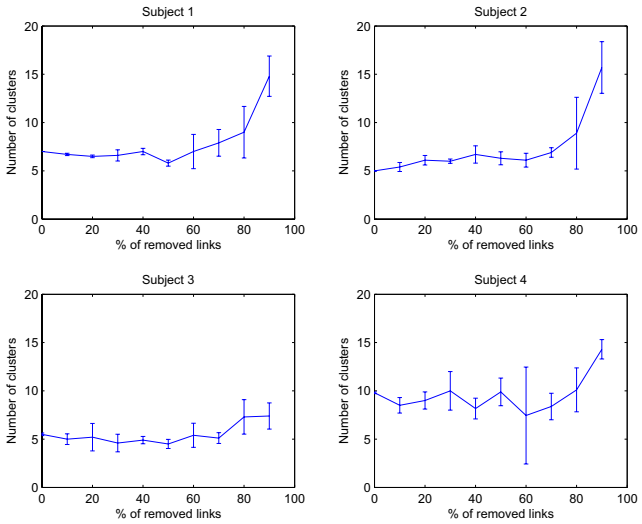
Figure 6: Time measurement of the group initialization experiment. For each scenario, each of the 4 subjects had to create a group using the three different selection modes: Choosing contacts from an alphabetic list, from the cluster view and based on recommendation.

In the beginning only few people are registered and many friendships links are missing. Our recommendation engine should also work in such environments. That is, it should yield good results even in an early phase of the service, and not require the social graph to be complete.

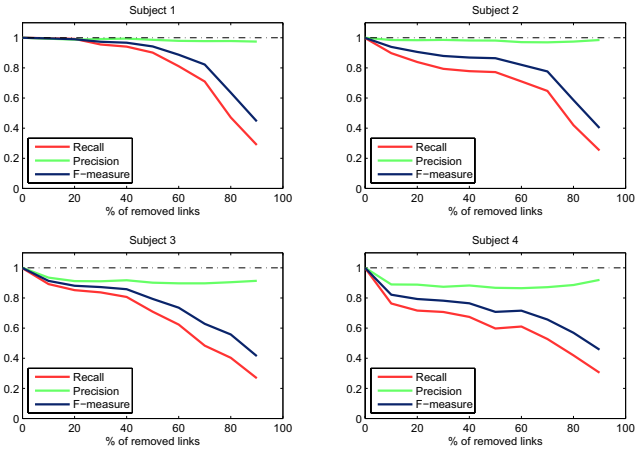
We thus also analyzed our recommendation algorithm's performance on degenerated networks, which are supposed to resemble the topological structure of a social services in an early stage. Two aspects have been investigated:

- **Missing friendships**: To reproduce the effect of missing friendships, we randomly removed links from the subjects' ego-graphs.
- **Missing users**: To investigate the effect of user sparsity, such as encountered shortly after launching a new services, we randomly removed nodes from the ego-graphs.

In both scenarios, the percentage of removed items (links or nodes, respectively), was increased from 0% to a total of 90% in 10% steps. After each step, the resulting graph was clustered. Possibly resulting single node clusters were ignored, as they are not relevant to the recommendation engine. To mitigate random effects, the entire procedure

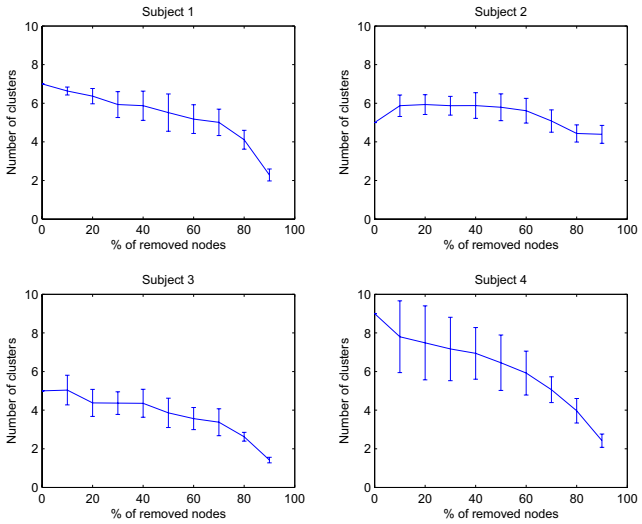


(a) Mean value and variance of number of detected clusters at each stage

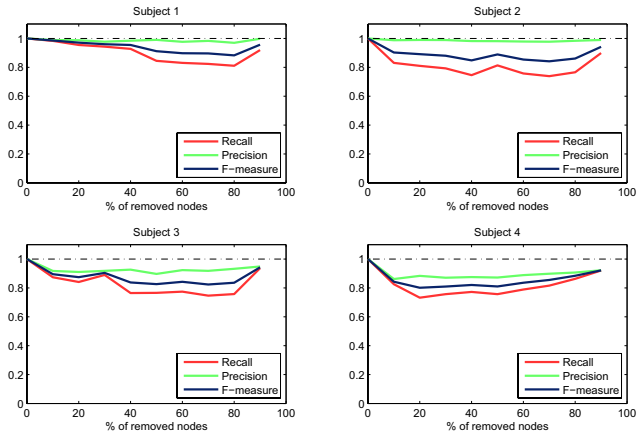


(b) Recall, precision and F-measure at each stage

Figure 7: Effect of clustering an ego-graph with missing friendship links. Links were removed randomly. Each stage was evaluated 30 times.



(a) Mean value and variance of number of detected clusters at each stage



(b) Recall, precision and F-measure at each stage

Figure 8: Effect of clustering a degenerated ego-graph with missing contacts. Nodes were removed randomly. Each stage was evaluated 30 times.

was repeated 30 times. For both scenarios we assessed the accuracy of the detected clusters. Moreover, we investigated the development of the F-measure, i.e., whether it remains stable or decays.

Figure 7(a) analyzes the effect of missing ties on the number of clusters. It plots mean and variance of the number of detected clusters at each step. The figure shows that the number of clusters increases with an increasing number of missing links. The reason for this behavior is that formerly densely connected nodes start to lose connectivity when links are removed. As a consequence, clusters fall apart into smaller pieces. Nodes do not get assigned to wrong clusters but remain together with other nodes of the same community. As a consequence, the precision of the newly generated clusters remains high. The recall, however, decreases rapidly when clusters split up, which also pulls the F-measure down. These effects are illustrated in Figure 7.

The second experiment addresses bootstrapping issues. Shortly after launching a new social service, the number of subscribers is typically low. Nevertheless, the service has to be usable in order to attract new users. Therefore, we investigate the effect of user sparsity on the performance of the clustering accuracy. The difference to the previous experiment is that nodes rather than links are removed from the graph. Removed nodes reflect an initiator's real life contacts that are not (yet) subscribed to the service.

The same measurements as in the first experiment were applied and evaluated. Figure 8(a) shows that the more nodes are missing, the less clusters are found. The reason for the decay of the number of estimated clusters is that the clusters dissolve since their nodes are not present anymore.[13] Figure 8(b) shows precision, recall and F-measure values. The F-measure is high regardless of the amount of removed nodes. Although the number of clusters decreases, the high precision value indicates an accurate clustering.

[13]Recall that single node clusters have been ignored.

Obviously, missing nodes do not significantly compromise the algorithm's performance. Consequently, recommendations based on an incomplete graph are still adequate.

We conclude that only a small subset of a user's real life ego-graph is required to estimate community affiliation among these contacts. Therefore, even with a small user base, Cluestr is able to provide acceptable recommendations. This helps to overcome the bootstrapping problem. However, contacts that are not registered to the service will never be recommended.

7. CONCLUSION

In this paper, we have proposed a contact recommendation mechanism which is designed to ease the initialization of group interactions from small devices. This kind of interaction will play an ever bigger role in the rapidly changing world of mobile communication that attracts more and more social services. Application scenarios have been sketched that underline these trends and demonstrate the importance of an efficient group initialization process. Moreover, we have integrated the proposed recommendation engine into a proof-of-concept application, and conducted preliminary experiments using real-world data. In particular, we have shown that:

- A user's ego-graph contains a significant amount of community information.
- This information can accurately be extracted by means of a state-of-the-art clustering algorithm.
- The extracted clusters can be used to recommend contacts that might fit into an existing group.
- This recommendation process can save a considerable amount of time when it comes to group contacts on a mobile device.

Moreover, we have demonstrated that the accuracy of the approach only moderately decreases if data gets sparse. Thus, our approach is applicable even in upcoming services that do not yet possess a large and stable user basis.

8. REFERENCES

[1] F. Bentley and C. J. Metcalf. Sharing motion information with close family and friends. In *CHI*, 2007.

[2] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *ESA*, 2003.

[3] S. Counts. Group-based mobile messaging in support of the social side of leisure. *Computer Supported Cooperative Work*, 16(1-2), 2007.

[4] G. Cselle, K. Albrecht, and R. Wattenhofer. Buzztrack: topic detection and tracking in email. In *IUI*, 2007.

[5] S. Farnham and P. Keyani. Swarm: Hyper awareness, micro coordination, and smart convergence through mobile group text messaging. In *HICSS*, 2006.

[6] S. Farnham, W. Portnoy, A. Turski, L. Cheng, and D. Vronay. Personal map: Automatically modeling the user's online social network. In *INTERACT*, 2003.

[7] M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 2002.

[8] S. Gregory. An algorithm to find overlapping community structure in networks. In *PKDD*, 2007.

[9] S. Gregory. A fast algorithm to find overlapping communities in networks. In *ECML/PKDD*, 2008.

[10] I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. In *IUI*, 2009.

[11] T. Hirsch and J. Henry. Txtmob: text messaging for protest swarms. In *CHI Extended Abstracts*, 2005.

[12] R. Kikin-Gil. Affective is effective: how information appliances can mediate relationships within communities and increase oneŠs social effectiveness. *Personal and Ubiquitous Computing*, 10(2), 2006.

[13] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis. Tastes, ties, and time: A new social network dataset using Facebook. com. *Social Networks*, 30(4), 2008.

[14] P. J. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. G. Terveen. Because i carry my cell phone anyway: functional location-based reminder applications. In *CHI*, 2006.

[15] M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), 2006.

[16] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23), 2006.

[17] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[18] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending modularity definition for directed graphs with overlapping communities. 2008.

[19] Nokia. plazes.com. www.plazes.com, 2009. [Online; accessed 13-January-2009].

[20] A. Oulasvirta, M. Raento, and S. Tiitta. ContextContacts: re-designing SmartPhone's contact book to support mobile awareness and collaboration. In *Mobile HCI*, 2005.

[21] P. Persson and Y. Jung. Nokia sensor: from research to product. In *Designing for User eXperience*. AIGA: American Institute of Graphic Arts New York, NY, USA, 2005.

[22] I. Smith, S. Consolvo, J. Hightower, J. Hughes, G. Iachello, A. LaMarca, J. Scott, T. Sohn, and G. Abowd. Social Disclosure of Place: From Location Technology to Communication Practice. In *Proc. Pervasive*, 2005.

[23] J. C. Tang, N. Yankelovich, J. Begole, M. V. Kleek, F. C. Li, and J. R. Bhalodia. Connexus to awarenex: extending awareness to mobile users. In *CHI*, 2001.

[24] M. von Arb, M. Bader, M. Kuhn, and R. Wattenhofer. Veneta: Serverless friend-of-friend detection in mobile social networking. In *WiMob*, 2008.

[25] S. Wasserman, K. Faust, and D. Iacobucci. *Social Network Analysis : Methods and Applications*. Cambridge University Press, 1994.