Automated Group Formation in Decentralised Environments

Hasini De Silva, Francois Carrez and Klaus Moessner
Centre for Communication Systems Research
University of Surrey
Guildford, UK
{h.desilva,f.carrez,k.moessner}@surrey.ac.uk

Abstract—Collaboration towards a goal involves groups of entities collectively possessing characteristics required to accomplish the goal. Facilitating collaborations in pervasive environments requires the automated formation of such groups. The group formation process is especially challenging in decentralised environments where there is no single central entity that can coordinate the formation process. It is also important that the group formation mechanisms are generic in nature so that they can be utilised in heterogeneous target environments regardless of their domain and requirements. This paper proposes a generic approach for automating group formation in decentralised environments.

Keywords—automated group formation; group collaboration; decentralised computing; pervasive environments

I. INTRODUCTION

With the emergence and popularity of pervasive computing, automation of collaborative tasks is becoming a reality. Automated formation of groups of entities capable of accomplishing the goals in these collaborative tasks is the first step in realising automated collaborations. Since the environment in concern is a pervasive environment, the collaboration tasks could be triggered anytime, anywhere with any type of entities. In other words, it is important that the Group Formation (GF) mechanisms are generic so that it can be adopted in any situation. Therefore, this paper investigates and proposes a generic solution for automating decentralised GF in pervasive environments.

Most GF approaches in the literature form groups based on member similarity [1][2][3]. Computer Supported Collaborative Learning (CSCL) is an area where criteria other than similarity are considered for the formation of learner groups. However, GF approaches in CSCL [4][5] are generally centralised. Due to the decentralised nature of pervasive environments, a single entity is usually neither capable of coordinating the GF process nor has a complete view of all the potential group members. In terms of decentralised GF, network clustering approaches [6][7] can be inspiring in the way they form into groups in the absence of centralised coordination. But, most such approaches focus on fairly simple GF criteria, as the aim is to partition a large network into manageable sized portions [8].

Most GF approaches are application specific [2][3][4][5], but MoPiDiG [1] is one approach that enables domain-independent GF, in which mobile hosts are grouped based on their profiles. It is also the closest approach found in the literature for the type of decentralised GF addressed in this work. This method achieves a global decentralised GF solution based on localised groups. However, GF criteria in MoPiDiG are restricted in the sense that GF is based on the similarity of member profiles. The objective of the work addressed in this paper is to propose a generic decentralised GF approach that does not impose restrictions on GF criteria so that it can be utilised to form groups regardless of the nature and the requirements of the target environment.

The rest of the paper is organised as follows: Section II describes the environment and requirements and Section III presents the proposed GF criteria specification and the proposed decentralised GF approaches. The performance of the approaches is discussed in Section IV in terms of simulation results, comparing the proposed approaches. A discussion is provided in Section V and the paper is concluded in Section VI.

II. PROBLEM SPECIFICATION

A. Environment

The environment considered in this work is represented as an undirected graph $H = (X, E)$, where the set of vertices $X$ represent the pool of individuals that are potential members of the formed groups, and the set of edges $E \subseteq X \times X$ represent communication links between individuals. An individual $x \in X$ has a set of neighbours $N_x = \{y : y \in X \land (x, y), (y, x) \in E\}$. It is noted that a neighbour of $x$ in this case is not necessarily physically close to $x$.

The individuals in $X$ are expected to form into groups. All possible groups are represented by the set $G$. Each individual $x \in X$ is mapped to a group $g \in G$ by the membership function $m : X \rightarrow G$. The set of members of a group $g \in G$ is represented by the set $M_g = \{x : x \in X, m(x) = g\}$.

Features are defined for the individuals by the use of a partial function $\nu_f : X \rightarrow V_f$ for each feature $f$, where $V_f$ is the value space corresponding to $f$.

In order to assign an aggregate feature value corresponding to a group $g$, a partial function $\nu_f^G : G \rightarrow V_f$ called the aggregate function, is defined for each feature $f$.

For each considered feature $f$, the feature distance between two values $v_1$ and $v_2$ in the corresponding value space $V_f$ is defined by the feature distance function given
by $d_f: V_f \times V_f \to [0,1]$. A constraint $c$ consists of a feature $f$ and a corresponding set of values $V_c \subseteq V_f$ and constrains the feature $f$ to the values in the set $V_c$. Satisfaction of a constraint by an individual is defined by the constraint satisfaction function, $s$, where, $s(c, x) = 1$ if $v_f(x) \in V_c$ and $s(c, x) = 0$, otherwise.

In the context of this paper, it is assumed that the set of individuals $X$ is fixed throughout where no additions or removals to or from $X$ occur and that the features $f_i$ defined on each $x$ are fixed for the entire period considered.

B. Requirements

The main goal is to enable the individuals to form one or more groups depending on GF criteria as described in a Group Formation Request (GFR) in a decentralised manner. The GF process should satisfy the following requirements: (1) Each individual $x \in X$ should only exchange messages with its set of neighbours $N_x$; (2) The GF approach should be decentralised; (3) An individual should be a member of at most one group; (4) The group formation process should terminate after a finite number of message exchanges; (5) At the end of the group formation process, each member of the formed groups must be aware of their groups and also the rest of the members of the group; (6) All formed groups should meet the criteria stated in the GFR.

III. DECENTRALISED GROUP FORMATION

A specification for GF criteria and mechanisms for decentralised GF is proposed in this section to meet the above requirements.

A. Group Formation Criteria

A group’s capability to accomplish a goal depends on its characteristics, which in turn depend on the features of group members. Therefore, two types of GF criteria that are built upon features of the individuals are defined.

Necessary Criteria: Necessary criteria are introduced to specify characteristics that are common to all group members. These criteria must be satisfied by all group members and can be defined either by setting a constraint on the allowed value for a feature or by specifying a set of features that should be similar among group members (definition methods are explained shortly).

Specialist Criteria: Specialist criteria are introduced to enable the formation of groups in which each member has a unique feature. They are defined using constraints, such that in the GF process each group has at least one member that satisfies each specialist constraint.

Two methods are proposed for defining the above criteria:

Constraint-based definition: In constraint-based definition, a criterion is defined using a constraint $c$, which is characterised by the set of allowed values $V_c$ for a particular feature $f$. This method can be used to define both types of criteria. For necessary criteria, all members of the group are expected to satisfy the necessary constraints ($C_n$), whereas for specialist criteria, at least one member of the group should satisfy each of the specialist constraints ($C_{sp}$).

Similarity-based definition: Feature similarity can be used to define necessary criteria, in cases where the actual feature values of the members does not matter; as long as the values are similar among members. A set of features, $F_s$, that are to be similar among group members and a threshold distance $d_{th} \in [0,1]$ which is used as a cut-off value for determining similarity are defined.

Therefore, a GFR includes: the set of similar features ($F_s$), the threshold distance ($d_{th}$), the set of necessary constraints ($C_n$) and the set of specialist constraints ($C_{sp}$).

B. Spanning Tree-based Group Formation (STGF)

This approach adopts a spanning tree structure used in [1], as a means of propagating information. Formation of a spanning tree requires each individual transmitting one round of messages to each of its neighbours. Once formed, information can be propagated through the network of individuals with each individual receiving only one message. An individual, on receipt of a GFR, creates a spanning tree and forwards it through the tree. Once this is received by all nodes, the leaf nodes check the criteria to determine whether or not they can initiate a group.

Initiating a group: An individual is able to initiate a group if all necessary constraints and at least one of the specialist constraints are satisfied.

If a leaf node $x$ is able to initiate a group $g$, Group Information (GI$_g$) is compiled. This contains information related to $g$ such as the associated GFR, members, the set of remaining specialist constraints ($C_{sp,g}$) which is recalculated after each addition to the group, necessary constraints ($C_{n,r}$), aggregated feature values of the group and information about similar features $F_s$. This information is sent to the corresponding parent nodes. Each non-leaf node merges all received information into a Combined Group Information (CGI) message, and tries to join a group depending on the satisfaction of constraints and the feature distance to the groups.

Feature distance: The process of merging or joining groups is based on the feature distances (between groups and between groups and individuals) with respect to the similar features $F_s$ contained in the GFR. For this purpose, for each feature $f \in F_s$, a feature distance function $d_f$ and an aggregate function $v_f$ (given in Section II.A.) is defined for each GFR by the target environment. For example, consider a numeric feature user preference that represents the numeric rank given by an individual for a particular item. A target environment may define the distance function by taking the numeric difference between two values and normalising it to the range $[0,1]$ and the aggregate function by taking the average. Allowing such application-specific definitions makes this approach generic, enabling the
1. Handle: GFR \( r \) received by individual \( x \)
   1.1. If initiator,
   1.1.1. Start spanning tree
   1.1.2. Forward GFR to child nodes
   1.2. Else if non-leaf node,
   1.2.1. Forward GFR to child nodes
   1.3. Else if leaf node,
   1.3.1. If \( r \in C_0 \), \( s(c, x) = 1 \) and \( 3c \in C_{ps} \), \( s(c, x) = 1 \),
   1.3.1.1. Create new \( IG_g \) and \( IG_g.add(x) \)
   1.3.1.2. Create new CGI and CGI.add(\( IG_g \))
   1.3.1.3. Forward CGI to parent
   1.3.2. Else,
   1.3.2.1. Forward empty CGI to parent
   2. Handle: CGI received by individual \( x \)
   2.1. If sender is child,
   2.1.1. Merge CGIs
   2.1.2. Forward CGI to parent
   2.1.3. If \( g \in C_0 \), \( s(c, x) = 1 \) and \( 3c \in C_{ps} \), \( s(c, x) = 1 \),
   2.1.3.1. If \( 3c \in C_{ps} \), \( s(c, x) = 1 \) and \( d(g, x) \leq d_{ps} \),
   2.1.3.1.1. \( IG_g.add(x) \)
   2.1.3.2. If \( g \) is not grouped,
   2.1.3.2.1. Create new \( IG_g \) and \( IG_g.add(x) \)
   2.1.3.2.2. CGI.add(\( IG_g \))
   2.1.4. Forward CGI to parent
   2.2. Else if sender is parent,
   2.2.1. Forward finalised CGI to child nodes

Figure 1. Algorithm for STGF

proposed GF functionality to be adopted universally regardless of the features considered. The method of definition of these functions, however, is out of the scope of this paper. Once these functions are defined for a given feature, the distance between two groups \( g_1 \) and \( g_2 \) is given by \( d_i(v_f, g_1), v_f(g_2) \) and the distance between a group \( g \) and an individual \( x \) is given by \( d_f(v_f(g), v_f(x)) \). The Euclidean distance metric is used to obtain the corresponding combined distances \( d(g_1, g_2) \) and \( d(g, x) \), over all features \( f \in F \).

The compiled CGI is then forwarded to the parent. This process is repeated until the information reaches the root node of the spanning tree, which then combines everything together and forwards the finalised CGI down the tree. The algorithm for this process is given in Fig. 1.

C. Flooding-based Group Formation (FGF)

This approach involves individuals flooding their neighbours with pieces of information. An individual “merges” each received new piece of information with existing information. If this triggers a change to the composition of a group, neighbours are updated by flooding the update. Flooding is continued until no updates are triggered by any individual, which means that an optimal solution has been reached.

When a previously unhandled GFR is received by an individual \( x \), a group \( g \) is initiated, if \( x \) satisfies the constraints (as in STGF described in Section III.B). In this case a \( GI_g \) message is forwarded to all neighbours. Otherwise, the received GFR is forwarded.

The received \( GI_g \) is merged with any matching existing groups. Then the individual considers joining group \( g \), depending on feature distances, or initiating a new group \( g' \) (see Section III.B). Any updates are notified by flooding the neighbours. Fig. 2 contains the algorithm for FGF.

D. Hybrid Group Formation (HGF)

This is a hybrid of the two methods described above. It follows the merge-and-update logic of FGF, but instead of flooding messages, uses a spanning tree as in STGF to forward updates to parent and child nodes.

E. Correctness of the algorithms

The satisfaction of the requirements stated in Section I.B by the proposed approaches is discussed in this section. Requirement 1 is met by all approaches, since in STGF and HGF, individuals exchange messages with a subset of their neighbours (i.e. parent and child nodes) and in FGF, individuals exchange messages by flooding their neighbours. FGF and HGF are decentralised, since all nodes make decisions independently; and STGF is decentralised, since there is no single entity responsible of decisions: each node’s decision depends on decisions of child nodes.

Therefore, requirement 2 is met by all approaches. Requirement 3 is met by STGF, since each node considers joining a group only once. FGF and HGF meet requirement 3, since an individual always checks whether it is already a member of a group, before joining one. In STGF and HGF, spanning trees are created by each node by forwarding one set of messages to their neighbours. STGF takes three passes of messages through the tree to achieve a solution. FGF and HGF propagate information until there are no more updates. Once one individual achieves an optimal solution, it replaces all sub-optimal solutions throughout the pool ensuring the termination of message exchange. Therefore, requirement 4 is met by all approaches. Requirement 5 is

1. Handle: GFR \( r \) received by individual \( x \)
   1.1. If has not handled \( r \) before,
   1.1.1. If not already a member of a group and
   \( \forall c \in C_0 \), \( s(c, x) = 1 \) and \( 3c \in C_{ps} \), \( s(c, x) = 1 \),
   1.1.1.1. Create new \( GI_g \) and \( GI_g.add(x) \)
   1.1.1.2. Store and forward \( GI_g \)
   1.1.2. Else,
   1.1.2.1. Forward GFR
   2. Handle: \( GI_g \) received by individual \( x \)
   2.1. If \( GI_g \) is not stored
   2.1.1. Merge \( GI_g \) with any matching stored \( GI_g \)
   2.1.2. If not already a member of a group and
   \( \forall c \in C_0 \), \( s(c, x) = 1 \) and \( 3c \in C_{ps} \), \( s(c, x) = 1 \),
   2.1.2.1. Create new \( GI_g \) and \( GI_g.add(x) \)
   2.1.2.1.2. Store and forward \( GI_g \)
   2.1.2.2. Else
   2.1.2.2.1. Forward \( GI_g \)
   2.1.2.2.1.1. Create new \( GI_g \) and \( GI_g.add(x) \)
   2.1.2.2.2. Store and forward \( GI_g \)
   2.1.2.2.3. If \( GI_g \) was modified,
   2.1.2.2.3.1. Store and forward \( GI_g \)
   2.1.3. Else if \( GI_g \) was modified,
   2.1.3.1. Store and forward \( GI_g \)

Figure 2. Algorithm for FGF
met in STGF, since the initiator compiles the final group information and forwards it through the tree to all individuals. Since message exchange is terminated only after all individuals achieve the optimal solution, requirement 5 is met in FGF and HGF. It is ensured that requirement 6 is met in all methods, since the composition of a group is altered only after considering the criteria.

IV. PERFORMANCE EVALUATION

To evaluate the performance of the GF approaches experimentally, the algorithms were run in an in-house simulator (details not included due to space constraints). The three approaches were compared against each other as well as against the approach presented in MoPiDiG [1]. However, since MoPiDiG does not support constraint based GF, in the test runs where the proposed approaches are compared against MoPiDiG, the GFRs are adapted such that they contain only similarity-based criteria, for which a numeric feature was used. In the test runs where the proposed approaches are compared against each other, GFRs with full-fledged criteria are used: a numeric feature for similarity and a nominal feature to define three specialist constraints. The tests were performed for pool sizes from 20 to 50. Larger pool sizes were not simulated, since the main aim of this evaluation is to compare the approaches rather than evaluate the scalability for larger pool sizes. The following paragraphs discuss various metrics used compare the different GF methods.

**Accuracy of the solution**: Two metrics are used to measure the accuracy of the solutions for the two forms of GFRs. The completeness index is formulated in such a way that it reflects the closeness of the members of the groups and the satisfaction of necessary and specialist constraints stated in the GFR. The higher the index, the more accurate the solution is. When the GFR is restricted to similarity-based criteria, the GF solution is essentially reduced to a clustering problem. To best portray the accuracy of solution, the Rand index [9], which is a measure of similarity of two data clusterings, is adopted. Each solution is compared against an optimal solution that is achieved by clustering the same pool of individuals in a centralised rather than decentralised manner. The closer the Rand index is to 1, the more similar the two clusterings.

**Message transmission**: The message transmission is evaluated in terms of the number of messages sent per node.

**GF time**: The duration from the receipt of the initial GFR by an individual to the completion of GF is considered as GF time or \( t_{GF} \). Although \( t_{GF} \) does not reflect the actual GF time due to the naive abstraction of communication functionality in the simulator, it is still a useful metric in comparing the GF approaches.

A. **Comparison against existing approach**

The proposed GF approaches are compared against that of MoPiDiG, in which case the GFR is configured to fit the problem addressed in MoPiDiG by limiting the GF criteria to similarity-based criteria.

Fig. 3 portrays that in terms of message exchange, STGF is better performing than MoPiDiG, while the performance of FGF and HGF are worse. STGF’s superior performance in this case is attributed to the fact that it takes only three passes through the spanning tree for achieving a solution. Since MoPiDiG nodes exchange additional messages among immediate neighbours in order to form localised groups, message transmission is higher than in STGF. FGF understandably performs worse in terms of message exchange, since it requires several rounds of messages to be propagated through flooding, before an optimal solution is reached. In this case, the probability of an individual receiving duplicate messages is very high. HGF has significantly better message counts since it uses the spanning tree to forward messages which prevents individuals from receiving duplicate messages, but is still
worse than STGF and MoPiDiG since it is built on the same principles as FGF where individuals constantly updates their neighbours during the GF process.

In terms of GF time, again STGF and HGF perform better than MoPiDiG as seen in Fig. 4. Although FGF is quicker in a pool size of 20 than MoPiDiG, for pool sizes from 30, it performs worse. STGF and HGF are quicker than MoPiDiG due to that fact that no additional time is spent on forming localised groups. However, FGF’s performance is impaired drastically with the increase of pool size due to the increase in sub-optimal solutions which gives rise to more rounds of messages. GF time in the other three approaches increases slightly with the pool size, as the time taken to build the spanning tree in each case increases with the pool size.

Fig. 5 compares the accuracy of the solutions in terms of the Rand index. All three proposed approaches seem to have a similarly high performance while MoPiDiG’s performance is not matched. The similarity of the performance of the proposed approaches here is due to the fact that the problem is reduced to a clustering problem which is handled similarly by all three approaches. MoPiDiG’s performance is worse since there is the possibility of individuals not being included in groups due to being unable of forming a localised group.

B. Comparison of the approaches against each other

To compare the GF approaches, a GFR in the full-fledged configuration which includes similarity-based as well as constraint-based GF criteria is used. From Fig. 6 and Fig. 7 it is evident that in terms of message exchange as well as GF time, the FGF method performs worse, which is similar to the findings in the previous sub-section. This is due to the fact that FGF requires several rounds of messages to be propagated before an optimal solution is reached. Furthermore, as the pool size gets larger this effect gets worse, since sub-optimal solutions increase, giving rise to more rounds of messages. GF time in STGF increases slightly with the pool size, as the time taken to build the spanning tree increases with the pool size.

To measure the accuracy in this case, the completeness index is used. It is seen in Fig. 8 that although STGF is more accurate for a pool size of 20, as the pool size increases, the performance falls below that of FGF and HGF. This impairment is due to the fact that STGF is more likely to reject an individual more suitable for a particular group at an early stage owing to its hierarchical nature. The accuracy of FGF is better than that of HGF, since the probability of including more suitable members in the groups is high due to the wider penetration of information achieved through flooding.

V. Discussion

According to arguments provided in Section III.E, it is seen that all of the GF approaches proposed in this paper meet the requirements. Results in Section IV.A portray that STGF is superior to MoPiDiG in all aspects when the GFR is configured so that the GF problem is reduced to a clustering problem, as addressed in MoPiDiG. Although MoPiDiG performs better in terms of message transmission than FGF and HGF, in terms of GF time, it is worse than HGF and has the worst performance in terms of the accuracy of the solution. In the comparison of the proposed approaches against each other in Section IV.B, STGF is still the best performing in terms of message transmission and GF time. However, the accuracy of STGF is impaired for large pool sizes. HGF seem to achieve an accuracy closer to that of FGF, which seems to be the most accurate solution, but in a much quicker time. Although messages sent per node in HGF are not as low as STGF, it is a significant improvement to the performance of FGF.

All in all, it can be seen that FGF is the approach that gives the most accurate GF solution. However, this accuracy comes at a cost of very high message transmission rates and
GF times. The hierarchical GF approach followed in STGF impairs its accuracy, but the use of a spanning tree has drastically reduced the message transmission and GF time. HGF, which is a hybrid approach between the other two, achieves an accuracy closer to FGF but at a much lower cost of message transmission and GF time, and seems to be the most favourable solution in general.

VI. CONCLUSION

In this paper, three methods are proposed for decentralised group formation based on features of individuals. The environment is assumed to be static, where no changes occur in the pool of individuals. Features of the individuals are taken as the basis to define GF criteria which are included in a group formation request (GFR). Necessary and specialist criteria are defined based on either constraints set on feature values or the similarity among group members. The consistency of the formed groups can be varied by choosing the appropriate method to define criteria. Constraint-based definition results in groups with members having feature values exactly as defined in the criteria, whereas similarity-based definition results in groups with members having similar values for the specified features. If a GFR is defined using only constraints, only one group that satisfies the constraints in the GFR will be formed, while the usage of only similarity-based criteria essentially clusters the pool of individuals into several groups with respect to the similar features. By using a combination of similarity-based and constraint-based criteria, a group that satisfies the constraints is formed for each similarity-cluster. This gives rise to the formation of several of groups equal to the number of existing clusters with respect to the similar features.

Three approaches are proposed for group formation: STGF, FGF and HGF, all of which meet the requirements of GF. Owing to the hierarchical nature of the STGF approach, it is best suited in situations where a quick and less message costly GF solution is sought as long as the accuracy is not too important. FGF’s merge-and-update logic with the use of flooding gives a very accurate but message and time costly solution and therefore, can be adopted in accuracy critical situations. HGF utilises the merge-and-update logic of FGF that yields an accuracy closer to that of FGF but has cut down the message exchange and GF time by the use of a spanning tree, giving the best all round performance.

The proposed approaches are generic in such a way that they are independent of the features that are taken as the basis for GF. Features of any form can be used as a basis for GF, as long as a distance function \(d_f\) and an aggregate function \(v_f\) can be defined for each feature by the target environment (definitions of these functions are out of the scope of this paper).

This paper assumes a static environment where no changes occur in the pool of individuals. However, dynamicity, such as the appearance or disappearance of individuals as well as changes to feature values, is present in reality. Work is in progress to study how this affects the formed groups and to define mechanisms to maintain the formed groups in the presence of dynamicity.

ACKNOWLEDGMENT


REFERENCES