Requests Management for Smartphone-based Matching Applications using a Multi-Agent Approach

Gilles Simonin and Barry O'Sullivan

Insight Centre for Data Analytics
Department of Computer Science, University College Cork, Ireland
{gilles.simonin|barry.osullivan}@insight-centre.org

Abstract. We present a new multi-agent approach to managing how requests are sent between users of smartphone-based applications for reaching bi-lateral agreements. Each agent has a selfish behaviour based on his preferences and an altruist behaviour with respect to the links between the agent and his neighbours. The objective is to maximise the likelihood of an acceptable match while minimising the burden on the users due to unnecessary messaging. We provide a dynamic algorithm using this architecture and we present an empirical evaluation with various mathematical models of users' behaviour and altruism. The evaluation shows that our approach can reduce the risks of rejections and the number of requests while increasing the likelihood of acceptable matches.

1 Introduction

Recently the use of smartphone-based applications has received attention due to the emergence of a number of new online optimisation problems that involve reaching bi-lateral agreements, such as in ride-sharing applications. The majority of such apps require sending a large number of requests or notifications to users. A typical goal of providers of these apps is to reduce the inconvenience to users by minimising the number of requests sent. The interaction with users represents the single greatest challenge in these applications. A social model/approach cannot be used for this problem because of the lack of information from the users. The system knows only the positive/negative answers from each user who received a request, and his waiting time limit. The system can only decide dynamically when it must send a new request. We can only simulate realistic behaviour of users to help the system in his decision.

Several robotics systems that propose an easy dialogue with, and a limited number of interactions for, each user have emerged in the literature. Many models and self-satisfaction architectures have also been developed and proposed for reactive robotic multi-agents systems [2, 4, 7]. These authors focus their models on communication and cooperation between learning situated agents. Robotic applications require distributed solutions and adaptive cooperation techniques. The agents learn to select behaviours that are well adapted to their neighbours'

activities. Multi-agent systems for solving complex combinatorial problems are often formulated as Distributed Constraint Optimisation Problems (DCOPs) [3, 5, 6]. These authors propose search algorithms for DCOPs that are analogous to a reactive multi-agent architecture. These methods and algorithms are not applicable in our case since we are concerned with predicting an appropriate time to send a request between users. For our problem, we only need the principle of distributed communication but not the cooperation part. Our agents will not act to help a global mission or achieve a goal. We are using a self-satisfaction architecture and a model of altruistic behaviour for the purpose of simulating the potential stress of each user and helping the system to decide when and whom to send a new request.

The remainder of the paper is organised as follows. In Section 2 presents our requests problem and how it relates to the Satisfaction-Altruism-based architecture for the multi-agent problem. In Section 3 we present this architecture for a multi-agent system, along with our hybrid model with the different concepts and functions. Section 4 focuses our study on the specific application of ride-sharing, where we propose a more detailed model for this specific problem. Finally, in Section 5 we present experiments with this specific model and compare the effect of varying the critical parameters of the model with an greedy system used.

2 The General Requests Problem

One of the advantages of smartphone-based applications is their ability to reduce the demands placed on users and their ability to reach decisions in a timely manner. In many applications (ride-sharing, on-line service between users, games, sales, etc.) users must find a bi-lateral agreement, e.g. a driver is happy to offer a ride to someone who is happy to accept that offer. Each user sends a request to other users and waits for a positive/negative response. In many applications this leads to a proliferation of notifications and requests sent to and by users. For this reason many automated systems that take decisions on behalf of users have emerged. In these systems, users can only initiate the agreement process, but they cannot control how an agreement is reached.

This requests problem can be modelled as a graph: the set of vertices represents users and we define an edge between two users if there exists a potential bi-lateral agreement, a deal, between them. A weight is assigned to each edge in order to quantify the quality of the potential deal. The edge weight encodes importance, but not likelihood of a match. The objective is to minimise the risk of rejection and the number of notifications/requests sent between users. Therefore, the goal is to select a maximum number of edges (deals) that maximises the sum of the weights. It is obvious that this selection is constrained by the structure of the problem and by the nature of the smartphone app in question. For example, in the ride-sharing problem, we want to select the potential riders in the same car. Therefore, we want three edges from a driver, if we want to fill the car, and one edge for a rider. From a deterministic global solution (see [9]), requests are sent by the system to each user. If a user does not respond quickly,

the system has to propose an alternative solution to another potential partner who is waiting to make a deal. However, this alternative partner could be a poor quality match in comparison to the first ones, thus it might be advantageous to allow additional time to the partners already contacted before sending a request to an alternative new partner. The question is when should the system send such a new request? The quality of the solution depends on the reactivity and the type of answers from users. This problem can be seen as an online one in which we need a good protocol to satisfy our objectives.

One can see the requests problem as a Satisfaction-Altruism-based architecture for multi-agent problems [1, 8] where each user is represented by an agent. Each agent has a personal behaviour based on his preferences and altruistic behaviour with respect to the links between the agent and his neighbours. Such a behaviour-based approach provides a basis to design each agent's actions over time. This perspective allows us to define agents that are able to evolve in dynamic and partially unknown environments.

3 A Multi-Agent Approach

The main principle of our approach is to model, in real time, a realistic behaviour of smartphone users and to manage the automatic systems which control the sending of notifications and requests. This architecture uses three concepts of agent satisfaction: the **personal satisfaction** which measures the level of satisfaction (e.g. stress, wait); the **interactive satisfaction** which describes the benefit of the interaction values between an agent and its neighbours; and the **altruistic reactions** which involve monitoring personal/interactive satisfaction between agents/neighbours and in transforming them into specific actions. When an agent perceives a prior signal of mutual interaction from a neighbour and possess a negative personal satisfaction, it can have an altruistic reaction in order to satisfy the protagonists (agent/neighbour) involved. The general architecture is presented in Figure 1.

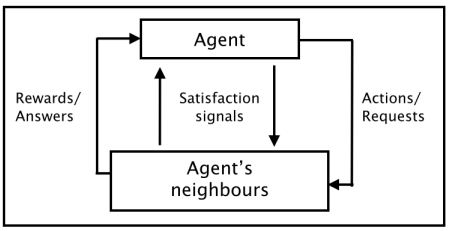

Fig. 1: Illustration of the requests managing architecture

In our problem, actions are the requests sent from a user to his potential partners during a given period of time. When an agent has a negative personal

satisfaction and perceives from one of his partner a signal of potential interaction, the user can change its action or start a specific one. The interactive satisfactions of neighbours are represented by the weight of edges between potential partners.

3.1 Personal Satisfiability

Each user u (agent) is active between a starting time $\boldsymbol{t_u^0}$ and an End of Time ($\boldsymbol{EoT_u}$) and is looking for one/several deal(s), or matches, with other users. A deal is represented as a mutual agreement to a request between the user and one of his partners. We want to simulate the fact that the closer a user gets to his time limit, the greater his stress. We can define a threshold under which the system may send a new request. Therefore, the personal satisfiability $\boldsymbol{Sat_u(t)}$ measures, at each moment in time, a simulated behaviour/stress of the user u. This function is bounded by $[-1, 1]$ and is decreasing exponentially according to a variation function $\Delta_{Sat_u}(t)$ at each elapsed time point, e.g. every second (see illustration Figure 2). In delimiting the threshold by 0, every time the function has a negative value, the user's stress is enough to let the system send a new request if there exists a potential partner. In this case, we increase the value $Sat_u(t)$ by 1.

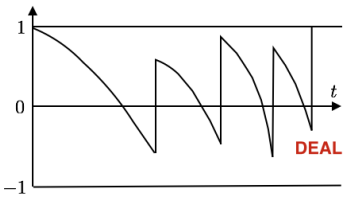

Fig. 2: Illustration of the personal satisfaction. Each spike represents the sending of a new request to a potential partner.

This variation function can change during the horizon time. For example, one can consider the very natural behaviour which is that the closer the user gets to time EoT_u, if a deal has not been made, the higher his stress level is likely to be. Likewise if he has already a deal and waits for another possiblity, his stress decreases more slowly. It is important to limit the decrease in the variation function, nevertheless in worse case the user's satisfiability will be always be equal to -1. This limitation depends on the application and the context; therefore we can define a maximum variation.

We also define a minimum waiting time duration between two new potential actions (sending a new request, that means a negative personal satisfaction). Let $\boldsymbol{T_{min}}$ be the minimum time for the personal satisfaction function to go from 1 to 0. We propose a particular variation function for the personal satisfiability of a user limited by the upper bound $1/T_{min}$.

Definition 1 (Variation Function) *The variation function $\Delta_{Sat_u}(t)$ for the personal satisfiability of a user is defined by:*

$$\Delta_{Sat_u}(t) = \min\left(\frac{1}{T_{min}}, \frac{\beta_u}{EoT_u - t}\right). \quad (1)$$

Here $\boldsymbol{\beta_u}$ is a positive stress impact factor that is dependant on EoT_u. The impact factor is an important parameter in our study. The higher its value, the faster the variation function will decrease. Observe that with $\beta_u \geq 0$ and $t_u^0 < t < EoT_u$, we can bound the variation function, $\Delta_{Sat_u}(t) \in [0, 1/T_{min}]$. From Definition 1 we can give the definition of personal satisfaction as follows.

Definition 2 (Personal Satisfaction) *The personal satisfaction $Sat_u(t)$ of an active user u at time t is defined by:*

$$Sat_u(t) = Sat_u(t-1) - \Delta_{Sat_u}(t). \quad (2)$$

A second function, which is somewhat more accurate, could take into account the number of deals the user is already involved in. In this setting his stress/behaviour score will decrease more slowly if he is already partially satisfied. Thus we propose to reduce the variation with the number of the user's deals $NbDeals_u$ as follows.

Definition 3 (Variation Function with Deals) *The variation function with deals $\Delta^D_{Sat_u}(t)$ for the personal satisfaction of a user is defined by:*

$$\Delta^D_{Sat_u}(t) = \min\left(\frac{1}{T_{min}}, \frac{\beta_u}{EoT_u - t} \times \frac{1}{NbDeals_u + 1}\right). \quad (3)$$

From the fact that $\beta_u \geq 0$ and $t_u^0 < t < EoT_u$, we can still bound the variation function by $\Delta^D_{Sat_u}(t) \in [0, 1/T_{min}]$. In Section 4 we will define a specific personal satisfaction function with different values to describe the impact factor β_u and T_{min} for the ride-sharing version of our general problem.

3.2 Interactive Satisfaction for Partners

Each active user u has a list of potential partners $P = \{p_1, p_2, \ldots, p_k\}$. For each pair $\langle u, p_i \rangle$, we have a weight representing the preference $w_{\overrightarrow{up_i}}$ from u to p_i. Thus we can define a list of weights for the potential partners: $W_{\overrightarrow{uP}} = \{w_{\overrightarrow{up_1}}, w_{\overrightarrow{up_2}}, \ldots, w_{\overrightarrow{up_k}}\}$. Weights are defined on the range $[0, MaxW]$, where $MaxW$ is the highest value. The higher the value of a weight, the better the chances of a deal. In the following, we will call **best partner** the one with the highest weight from the list of the unrequested partners.

The interactive satisfaction $\boldsymbol{IntSat_{up_i}(t)}$ measures, at every moment in time, the potential interaction between user u and his potential partner p_i during the user's time window $]t_u^0, EoT_u[$. This function is bounded by $[0, MaxW]$ and is decreasing differently according to a variation function $\Delta_{IntSat_{\overrightarrow{up_i}}}(t)$ at each elapsed time period, say every second (see Figure 3).

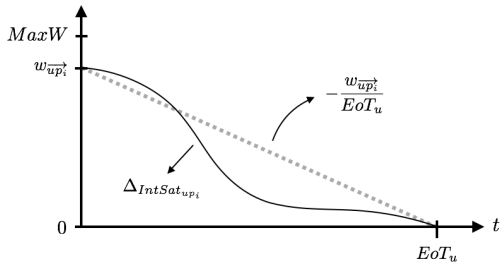

Fig. 3: Illustration of the interactive satisfaction function.

According to the satisfaction of u and his partners, $\Delta_{IntSat_{\overrightarrow{up_i}}}(t)$ will change over time. At every time step, this variation value decreases (resp. increases) if a request has been already sent to p_i (resp. not sent):

- In the decreasing case, the interactive value is decreasing from t_u^0 to EoT_u according to faster/slower variations around the constant decreasing ratio $\frac{w_{\overrightarrow{up_i}}}{EoT_u}$.
- In the increasing case, the interactive satisfaction remains at the same value or can increase slowly according to the partner's personal satisfaction. If his personal satisfaction is negative, the interaction value will increase. The increasing part of the variation function for the interactive satisfaction is upper bounded by a value in $[0, \frac{1}{EoT_u}]$.

Definition 4 (Variation Function for Interactive Satisfaction) *The variation function for the interactive satisfaction between an user u and his partner p_i is defined by:*

$$\begin{aligned}\Delta_{IntSat_{\overrightarrow{up_i}}}(t) = x_{\overrightarrow{up_i}} \times \left(\frac{w_{\overrightarrow{up_i}} \times \beta_{p_i}(t)}{EoT_u}\right) \\ + [1 - x_{\overrightarrow{up_i}}] \times \left(\frac{\min\{0, Sat_{pi}(t)\}}{EoT_u}\right)\end{aligned} \tag{4}$$

$$\text{where } x_{\overrightarrow{up_i}} = \begin{cases} 1, & \text{if a request has already been send to } p_i, \\ 0, & \text{if no request has not been send to } p_i. \end{cases}$$

and $\boldsymbol{\beta_{p_i}(t)}$ is a positive impact factor of stress depending of the partner's personal satisfaction and the time. The higher its value, the faster the variation function will decrease. From (4), we can give the definition of the interactive satisfaction:

Definition 5 (Interactive Satisfaction) *The interactive satisfaction $IntSat_{\overrightarrow{up_i}}(t)$ of an active user u at time t is define by:*

$$IntSat_{\overrightarrow{up_i}}(t) = IntSat_{\overrightarrow{up_i}}(t-1) - \Delta_{IntSat_{\overrightarrow{up_i}}}(t). \tag{5}$$

A second, more accurate, function could take into account the number of deals already owned by the user or his partners. His/their stress/behaviour will decrease/increase slower if he is already partially satisfied. Thus we propose to reduce the variation based on the number of the user's deals $NbDeals_u$ ($NbDeals_{p_i}$ for the partner p_i) already in place.

Definition 6 (Variation Function with Deals) *The variation function with deals for the interactive satisfaction of a user u and his partner p_i is:*

$$\Delta^D_{IntSat_{\overrightarrow{up_i}}}(t) = x_{\overrightarrow{up_i}} \times \left(\frac{w_{\overrightarrow{up_i}} \times \beta_{p_i}(t)}{EoT_u[NbDeals_u + 1]} \right) + [1 - x_{\overrightarrow{up_i}}] \times \left(\frac{\min\{0, Sat_{pi}(t)\}}{EoT_u[NbDeals_{p_i} + 1]} \right). \quad (6)$$

In Section 4 we will propose a specific definition of the impact factor $\beta_{p_i}(t)$ for a ride-sharing version of our general problem.

3.3 Altruistic Reaction

According to the updated values for the personal/interactive satisfactions, we can take into account the level of user stress and define a protocol to allow agents to modify their actions. In our problem, the system is authorised to send a new request from a user to his partners when specific conditions are satisfied. We define two conditions required to allow at time t a new request to be sent from a user u:

- The value of the personal satisfaction of u is negative.
- One of his unrequested partners has an interactive satisfaction higher than the current highest one. Specifically, the weight $w_{\overrightarrow{up_i}}$ increased enough and is now higher than the current highest weight from requested partners.

If both conditions are respected, the system is authorised to send a new request. The first condition allows to minimise the delay between two messages. Indeed, when a notification is sent the system increases $sat_u(t)$ by 1. The second condition allows more time to the best partners who might answer positively to the user u.

We are interested in finding a global solution over all users in our graph. When the system receives an authorisation to send a new request for a user u (vertex) to p_i, that means the edge $\{u, p_i\}$ can be selected in the global solution. If the computation of a global solution contains this edge, the system can send a new request to p_i from u.

From the point of view of the model, the system gives a reward to $Sat_u(t)$. This bonus can depend of the number of deals required by the user $MaxNbDeals_u$, and thus $Sat_u(t)$ increases by $1/MaxNbDeals_u$.

3.4 The Algorithm

We present the algorithm that monitors a user and his partners using the three multi-agent concepts, presented above, over time in order to reduce the number of requests sent while also minimising the risk of rejection (requests sent to partners with too low a weight).

```
   Data: A user u, his partners list P = {p_1, ..., p_k},
   His bounds t_u^0 and EoT_u, and a boolean Change
 1 begin
 2 |  Request sent to the best partner;
 3 |  while t_u^0 < t < EoT_u do
 4 |  |  Change ← False;
 5 |  |  while Not Change do
 6 |  |  |  Values updated;
 7 |  |  |  if Negative answer received then
 8 |  |  |  |  Remove the partner;
 9 |  |  |  if Positive answer received then
10 |  |  |  |  if Deal accepted by u then
11 |  |  |  |  |  Sat_u(t) ← 1;
12 |  |  |  |  |  NbDeals_u ← NbDeals_u + 1;
13 |  |  |  |  |  if u fully satisfied then
14 |  |  |  |  |  |  End of process;
15 |  |  |  |  end
16 |  |  |  |  Remove the partner;
17 |  |  |  if Altruism reaction for one partner p_i then
18 |  |  |  |  Change ← True;
19 |  |  |  end
20 |  |  |  t ← t + 1;
21 |  |  end
22 |  |  New Global Solution with 1 more edge from u;
23 |  |  if the edge {u, p_i} is selected in the solution then
24 |  |  |  Send a new request to p_i;
25 |  |  |  Sat_u(t) ← Sat_u(t) + 1;
26 |  |  end
27 |  end
28 end
```

Algorithm 1: Monitoring of requests sending for a user u.

Algorithm 1 comprises a while loop inside another general loop. At Line 3, we start a general loop on the time windows of user u. This *while* loop represents the active time of the agent/user u, during this time interval the system monitors the personal/interactive satisfaction functions.

In Lines $5-21$ we are in the Decision-Action loop. This *while* loop involves first computing and updating, after every time unit, the user's personal satisfaction Sat_u and the interactive satisfactions of his partners. Second we check

every case when a partner answers negatively or positively. If we have a positive answer and a deal with the user, it is necessary to increase values and to check if the user is fully satisfied. If he need more deals, we continue the process, else the system stops the monitoring. Third we test if the required conditions to send another request are satisfied, that means the personal satisfaction is negative and there exists a partner who has not yet received a request and has a weight higher than the best weight from partners already requested.

In Lines $22 - 26$ we compute a new global solution with the new constraints and we send a new request if it's relevant. The new request leads to a lower level of stress for the user and to an increase in $Sat_u(t)$.

An example of the algorithm in operation is presented in Table 1 for a user u with three partners $\{p_1, p_2, p_3\}$ during the time window $[0, 1500]$. The ending of the process can advance if the user is fully satisfied. We monitor user behaviour every five minutes, with the first request sent to p_1 with the highest weight. After $900s$, the required conditions are satisfied to send a new request to the new best partner. At $1320s$, the request by the partner p_1 is declined and leads two actions: p_1 is removed from the list of potential partners and a new request can be sent if the conditions are satisfied. At $1380s$, p_2 answer positively to u. If u accepts the deal and is fully satisfied, the process stops and we remove him.

Table 1: Example of the system between a user u and three partners. The boxes represent the sending of a request.

Time	$w_{\overrightarrow{up_1}}$	$w_{\overrightarrow{up_2}}$	$w_{\overrightarrow{up_3}}$	Sat_u
$t = 0s$	10	5	3	0
$t = 600s$	7.458	6.458	4.458	-0.49
$t = 900s$	6.09	7.091	5.091	$-0.836 \rightarrow 0.164$
$t = 1200s$	4.53	5.53	5.53	-0.406
$t = 1320s$	NO$\rightarrow 0$	4.855	5.655	$-0.721 \rightarrow 0.279$
$t = 1380s$	0	YES	5.342	$0.062 \rightarrow 1$

4 The Ride-Sharing Case

In this section, we focus on the specific case of our problem where users are drivers and riders, and where the smartphone-based application is a ride-sharing one providing an automatic system to users to propose matchings through bilateral agreements. Each driver can have three riders in his car, and the system tries to find a match for each user u on the time windows $[t_u^0, EoT_u]$. Using data collected from the application users and their comments, we provide the best functions representing their potential stress and behaviour. The system automatically computes the matching and request messages. Users do not know what the system is doing (sending requests to their partners). Therefore, we cannot model and represent their true behaviour and stress. We use this partial model to create a good protocol to the system to know what to do and when.

4.1 Greedy model

Previously on the application, the system was using a greedy strategy consisting in sending repeatedly more requests to other partners if the first ones do not answer. This method provides solutions for each user but the quality can be poor, and the number of requests sent increases exponentially. For a large number of users, it can be critical in terms of communication and management. This system does not take into account the quality of the weights. If none of the first partners answers quickly, a new one can be selected from a new request and his fast positive answer can lead to a matching of bad quality (low weight). In the experimental section, we will use the minimal delay required to send a new request in the greedy system as a lower bound for the value T_{min}.

4.2 Multi-Agent Model

We developed a multi-agent model for the ride-sharing requests problem as presented in the previous section. As one can see in the algorithm, the decision to send a new request depends on the values of the personal and interactive satisfaction functions. We define a specific function for each satisfaction for the ride-sharing version. Their update leads to an altruistic reaction from users to their partners. This section focuses on the specificities of these functions and on their implications for the objectives to control the risk of rejection while minimising the number of requests. In the following, we present for each function the different parameters which will vary in the simulations.

4.3 Updated Values: Personal Satisfaction

In Definition 1 we gave the definition of the personal satisfaction variation $\Delta_{Sat_u}(t)$ for a user u (and the deals version in Definition 3). This function depends on a stress impact factor β_u which depends on EoT_u. The higher its value, the faster the variation function will decrease.

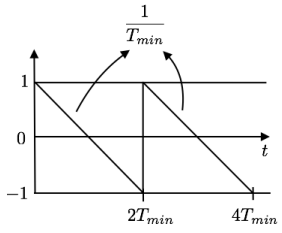

Fig. 4: Illustration of the lower bound.

A way to compute this impact factor involves choosing the number of times that we want the function to arrive at -1. For this, one can imagine that if

every time the function arrives at -1 we add 2, we could compute the number of times the personal satisfaction function is reaching -1. Using the lower bound variation $-1/T_{min}$, we can compute this number from EoT_u. From the fact that we need $2 \times T_{min}$ to decrease the function by -2 (see Figure 4), and that the personal satisfaction function is reaching -1 at least k times, we have the following definition.

Definition 7 (Impact Factor) *The impact factor β_u is define by:*

$$\beta_u = \max\left(1, \frac{EoT_u}{2 \times k \times T_{min}}\right). \tag{7}$$

From Definition 7 one can see that the only parameters that vary for the personal satisfaction function (with or without the deals) are k and T_{min}.

4.4 Updated Values: Interactive Satisfaction

In Definition 4 we defined the interactive satisfaction function $\Delta_{IntSat_{\overrightarrow{up_i}}}(t)$ for a user u and his partners (and the deals version in Definition 6). This function depends of a stress impact factor $\beta_{p_i}(t)$ which depends on the user u and his partner p_i. The higher its value, the faster the variation function will decrease. We tried several types of behaviour model and functions for the interactive satisfaction variation, and we kept the most relevant according to our problem.

Definition 8 (Ride-Sharing Requests Problem) *In the Ride-Sharing requests problem, the stress impact factor depending on the user u and his partner p_i is equal to $\beta_{pi} = \max\{0, 1-Sat_u-Sat_{p_i}\} \in [0,3]$. The greater the level of stress, the faster other partners will receive a request. Therefore, the interactive satisfaction varies according to the personal satisfactions from protagonists at rate*

$$-\frac{w_{\overrightarrow{up_i}} \times (1 - Sat_u - Sat_{p_i})}{EoT_u}.$$

5 Empirical Study

As one can see in Algorithm 1, the decision to send a new request depends on the values of the personal and interactive satisfaction functions. Their update leads to an altruistic reaction from users to their partners.

For the evaluation, we have been working with an industry partner in the area of ride-sharing. Our model was proposed to them. They implemented it in full, except for the notion of *personal satisfaction*. They deployed the application over a number of days to measure the impact on system performance. The results were very good, significantly reducing the number of rejections and the number of requests sent to users. However, we explained to them the necessity in also using the personal satisfaction parameter to ensure their service could scale as the number of uses increases over time.

We do not present this evaluation in the paper because (a) of confidentiality issues, and (b) because the number of users using the apps were not large enough to see extreme cases and prove our model. This is why we rely on simulation on huge numbers of users and varying several parameters.

Our experiments focus on the specificities of these functions and on the objectives to control the risk of rejection while minimising the number of requests. In the following, we present the different parameters which will vary in the simulations.

5.1 Variation of Parameters

As seen in Definition 7, we define a specific function to describe the personal satisfaction. This function depends on three parameters k, T_{min} and EoT_u. By definition, the higher the parameters T_{min} and k are, the longer the time required to send a new request. We focus our simulation around these three parameters, in order to identify the cases where the reduction of the number of requests or rejections is the most visible. The variations are the following:

- k will vary between 2 and 6. It represents the number of time that the personal satisfaction is decreasing by -1 at rate $-1/T_{min}$ between t_u^0 and EoT_u;
- T_{min} will vary between $300s$ and $600s$. In the greedy system, the time between each new request was $300s$. From this information, we put this value as a lower bound and to observe its impact on our model by increasing it;
- For each user, the horizon EoT_u will take a commun value of $3600s$, $7200s$ or $10800s$. An analysis at $1h$ can show the limit of our technique if there are not enough answers, whereas at $2/3h$ the system should be more robust;
- The density of the graph will also vary to check the impact of a large number of partners. We will focus our simulation on a small density of 30% or a larger one of 60%.

5.2 Results and Discussion

In Figure 5, we present the simulation results between the **Greedy system** where requests are sent every T_{min} minutes to a new partner and our **Multi-Agent model** (M-A). We considered randomly generated graphs and simulated our on-line protocol to manage the sending of requests, then compared the results according the different parameters of the functions introduced in this paper. We computed the percentage of benefit/loss between the two systems. All experiments were run on Intel quad-core $i7$ running Mac-OS X 10.9.5 with 16GB of RAM. We present for each instance (depending on a small number of critical parameters) the number of rejections, requests sent, sum of weights for deals selected and the number of satisfied users.

One can see quickly that the number of requests sent decreases strongly. For each instance, there are in average a reduction of 700 requests sent in the new

Inst.	EoT	T_{min}	k
I1	3600	300	2
I2	3600	300	6
I3	3600	600	2
I4	7200	600	6
I5	7200	300	2
I6	7200	300	6
I7	7200	600	2
I8	7200	600	6
I9	10800	300	2
I10	10800	300	6
I11	10800	600	2
I12	10800	600	6

Table 2

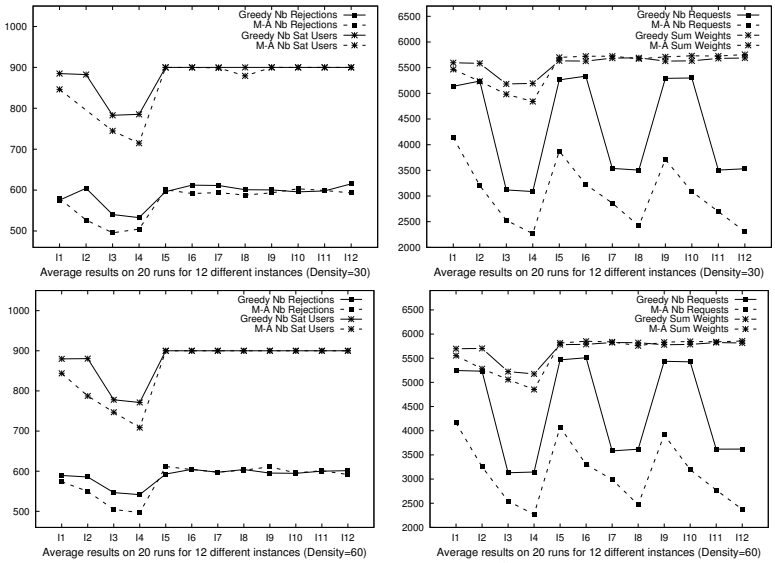

Fig. 5: Comparaison between the Greedy model and the Multi-Agent one using different values of k, T_{min}, EoT and density

model. One can observe that in each instance where we increase the value of T_{min} or k, the number of rejections is reduced in the new model.

The good news is that our approach seems robust in terms of its ability to reduce the number of requests sent in all cases. When the number of satisfied users is close between the two versions, the sum of weights is always better for our system. But for smaller values of EoT_u, the number of satisfied users is low for our system, since it does not have enough time to find a matching to everyone. This leads to a reduction of the number of deals obtained, and thereby of the sum of weights. By computing a ratio between the number of deals realised and the sum obtained, the matching obtained has a low level of risk as shown by the average value (weight) of each deal done for our system. Note that in real-world settings, users will start the application most often and the matching process three or four hours in advance. Finally, as the graph density increases the performance of our proposed method dramatically increases and describes well what we shown previously.

6 Conclusions

We have presented a new multi-agent approach to managing the requests made between users of applications for reaching bi-lateral agreements. The objective is to maximise the likelihood of acceptable matches while minimising the burden on the users due to unnecessary requests being sent. We presented a general model for this kind of smartphone-based matching problem, with two specific functions

to describe the satisfaction/stress behaviours and to limit spamming. We also defined and experimented with a specific model for the Ride-Sharing Requests Problem. Our results showed that this system succeeded in significantly and robustly reducing the number of requests sent even with large variation between the parameters. Their increase leads to a reduction the number of rejections and an increase in the sum of weights in the solution.

7 Acknowledgement

This paper has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 and from industry partner Carma (`https://www.gocarma.com`).

References

1. Balch, T., Arkin, R.: Communication in reactive multi-agent robotic systems. Autonomous Robots 1, 27–52 (1994)
2. Chapelle, J., Simonin, O., Ferber, J.: How situated agents can learn to cooperate by monitoring their neighbors' satisfaction. ECAI pp. 68–72 (2002)
3. Grinshpoun, T., Grubshtein, A., Zivan, R., Netzer, A., Meisels, A.: Asymmetric distributed constraint optimization problems. Journal of Artificial Intelligence Research 47, 613 – 647 (2013)
4. Hilaire, V., Gruer, P., Koukam, A., Simonin, O.: Formal driven prototyping approach for multiagent systems. International Journal of Agent-Oriented Software Engineering 2(2), 246–266 (2008)
5. Hirayama, K., Yokoo, M.: The distributed breakout algorithms. Artificial Intelligence 161(1-2), 89 – 115 (2005)
6. Maheswaran, R.T., Pearce, J.P., Tambe, M.: A family of graphical-game-based algorithms for distributed constraint optimization problems. Journal: In Coordination of Large-Scale Multiagent Systems pp. 127 – 146 (2006)
7. Mataric, M.J.: Behaviour-based control: Examples from navigation, learning, and group behaviour. Journal of Experimental & Theoretical Artificial Intelligence 9(2-3), 323–336 (1997)
8. P. Lucidarme, O.S., Liégeois, A.: Implementation and evaluation of a satisfaction/altruism based architecture for multi-robot systems. IEEE International Conference on Robotics and Automation 1, 1007–1012 (2002)
9. Simonin, G., O'Sullivan, B.: Optimisation for the ride-sharing problem: a complexity-based approach. ECAI pp. 831–836 (2014)