Supporting Collaborative Activity

Master Thesis

By: Gilad Armon-Kest
Advisers: Prof. Sarit Kraus,
Dr. Gal A. Kaminka

Submitted in partial fulfillment of the requirements for the Masters degree in the
Department of Computer Science Bar-Ilan University
Ramat Gan, Israel
2007

Abstract

This thesis has two parts. The first part presents SharedActivity, a model for collaborative agents acting in a group. The model suggests mental attitudes for agents with different levels of cooperation and allows modelling of groups in which members are motivated to increase individual benefits. Unlike previous models, SharedActivity is suitable also for groups that do not have a joint goal. The model defines key components of loosely cooperative activity and provides a platform for developing tools to support such activity. We have studied the behavior of the model in a simulation environment. Results show how the benefit attained by cooperation is influenced by the complexity of the environment, the number of group members as well as the social dependencies between the members. The results demonstrate that the model covers social behavior both in settings previously addressed, as well as novel settings.

The second part presents an algorithm for solving the problem of iterative search in a closed group. Our solution takes into account the load on agents and the agents' willingness to help other agents. It also manages reciprocity between agents. The proposed algorithm supports limited-resource platforms. We evaluate the behavior of our algorithm in a simulation environment and compare it to the random walk algorithm. Results show an advantage for our algorithm regarding the random walk algorithm in most environmental settings. These advantages are expressed in retrieving times of wanted objects and fairness in task distribution.

Acknowledgments

This dissertation could not have been initiated, conducted and completed without the invaluable guidance, inspiration, and support of my wonderful advisors, Sarit Kraus, Gal Kaminka and Meirav Hadad. Incessantly, they were patient and open towards all my Ideas and were willing to engage in them. By working with them, I accumulated vast knowledge, ideas, techniques and how to do a research.

I would like to mention the great support of everyone at the MAVERICK group who kept me company in my frequent visits there. I enjoyed many hours discussing research problems with a large group of people, in particular: Natalie Fridman ,Nirom Cohen-Nov-Slapak ,Ari Yakir, Efi Merdler, Victor Shafran, Meir Kalech ,Yehuda Elmaliach, Sarah Simani, Tom Shpigelman, Yael Termin, Avi Rosenfeld. These discussions often produced new ideas and helped clarify various technical points. I gratefully acknowledge Claudia Goldman-Shenhar and Vlad Luzin for their assistance in developing, debugging and executing the basic simulation software provided by STRI. Lastly and most importantly, I am forever indebted to my family who has been my source of strength throughout my life: my brother Shahar, my sisters Enat and Ayelet, my parents Shlomit and Uzi for always being there.

Last but not least, I thank my wife for always being there and for her help behind the curtains.

This research was supported in part by a research grant from Samsung Telecommunications Research, Israel (STRI), and by research grant #1357/07 from the Israel Science Foundation (ISF).

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of Bar-Ilan University.

Contents

List of Algorithms

List of Tables

List of Figures

Part I

The SharedActivity Model of Group Activity

Chapter 1

Introduction

Psychologists often classify groups according to the goals for which the groups are formed. Two common types are [22] task groups, and treatment groups. A task group is formed to accomplish a joint goal and thus the benefit of each member is immediately linked to the success of the joint task. In contrast, a treatment group is formed where the purpose of the members is to meet individual needs, but some cooperation is helpful. A treatment group is formed as a result of sharing common resources, situations, experiences, etc. Examples include: Students sharing the same lab, partners sharing the same apartment, researchers of the same research field, and a group who shares the same experience on a tour.

Previous work [13, 11, 6] has proposed well-grounded and explicit models of problem solving by distributed systems that work together in task groups. These models address essential characteristics of cooperative work and support the design and construction of collaborative systems. Using such models for multi-agent systems enhances the cooperation capabilities of the individual actors working on a joint goal (e.g., [9, 20, 16, 10].) A collaborative problem, in those models, is one in which various participants work jointly with each other, performing a task together or carrying out the activities needed to satisfy a joint goal.

Hence, previous models, concerning only task groups, are not applicable to support collaborative work in treatment groups, in which members are motivated to increase their own benefits, yet may still benefit from working with others. Consider, for example, an agent searching for information for its own goals, and during the searching process discovers information which interests another agent. By notifying the other agent about this information, it reduces the other's search costs. Repeated and mutual interactions of this type will increase both parties' utilities. Yet existing collaborative models (e.g. [6]) do not provide guidelines for the agents, since they do not have a common goal. Thus a new model is needed to support the development of agents capable of participating in treatment groups.

In this part, we will present the SharedActivity model to support both task

and treatment groups. The goal of the development is to provide a new and better platform for building tools with a high degree of flexibility to support collaborative activity. This model is novel and is able to contribute to group behavior which has not been previously modeled.

We investigated the behavior of our model empirically by applying it in a simulation of a small group visiting a museum. Our experiments test two points. One is how various environmental settings influence the agents' benefits when they are engaged in task group or a treatment group, and when they are not active in a group. The second is how the level of cooperation among members is influenced by environmental settings.

Chapter 2

Background

The SharedActivity definition in the next section, is based on studies of human groups. Toseland and Rivas [22] compare features of treatment and task groups. The comparison is presented in Table 1. As shown in this table, members of the treatment groups are bonded by their common needs and common situations, while members of the task group create a common bond by working together to accomplish tasks. In treatment groups, roles are developed through interaction among members. In task groups, roles are also frequently assigned by the group. Communication patterns in treatment groups are open and the members are usually encouraged to interact with one another. The communication patterns of task groups are focused on a particular task. Treatment groups are often composed of members with similar concerns. Task groups are frequently composed of members with the necessary resources and expertise to achieve the group's joint goal. Finally, the criteria to evaluate success differs between treatment and task groups. Treatment groups are successful to the extent that they help members meet their individual goals. Task groups are successful when they accomplish group goals such as generating solutions to problems. The SharedActivity model covers, in difference's levels, all of these features.

	Task groups	Treatment groups
Bond	a joint task or goal	common needs or situations
Communication	patterns focused on the particular task	flexible
Roles	frequently assigned by the group	evolve through interaction
Composed of members with:	the necessary resources and expertise to achieve the group's joint goal	similar concerns
Evaluating success	achieving group goals	the extent that they help members meet their individual goals

Table 2.1: Differences between groups

Previous models for supporting groups have been based on features of task groups, in particular for supporting teamwork. Levesque et al. [13] suggest concepts of joint commitment and joint intentions and study the ways in which they relate to individual commitments of group members. They address the need for agents to inform each other whenever they drop a joint commitment. These types of commitments are essential when satisfying a joint goal which determines a group's success.

The SharedPlan model [6] take the advantage of a group commitment to provide collaborative planning processes for achieving the joint goal. These include processes for how to perform activities and allocate tasks, as well as processes for coordination. However, the focus of the process is on the team goal, not individual goals.

Kinny et al [11] proposed explicit model of problem solving by distributed systems which work together in task groups. This model addresses essential characteristics of joint plans for teams of agents. Nevertheless in this model there is one plan that common to all team members that typical to task group.

An additional example of model that support task group is Joint Responsibility model of Jennings [9]. This model based on joint intentions. Group of agents in this model must satisfy certain conditions to achieve success. Yet, in ShardActivityt there is not requirement for such obligation.

BITE [10] architecture for multi-robot teamwork provides services for automated collaboration. This approach for flexible teamwork focuses on synchronization and task allocation. It enhances the cooperation capabilities of the individual actors working on a joint goal and therefore suitable only for task groups.

Recent work [19] has proposed model in which agents are allowed to change their level of cooperation over time as a function of their environment. In these model, the agent's cooperation measure depends on its personality and past experiences, as well as on the cost of helping. These work investigated the tradeoff between selfishness and helpfulness in environments in which agents are uncertain about the cooperative nature of others in the system. Nonetheless, it dose not provide a model to support a group formation by such agents. The model proposed in this paper deals with agents who act in a group but are able to adjust their cooperation level to the environment over time.

Chapter 3

The SharedActivities Model

The SharedActivities model is intended to be used in guiding the design of agents, by providing a specification of the capabilities and mental attitudes that an agent must have when it works as a part of a group. To motivate the discussion, we start with an informal example of a small group visiting a museum. The model was tested in a simulation of this task (see experiments section). We refer to this example throughout the paper.

3.1 An Example of a Collaborative Activity

Our example is motivated by the PEACH technology for museum visits. PEACH [8] offers adaptive multimedia presentations as the visitor moves around in a museum. One specific challenge in this project is to develop technology that supports group visits to the museum. In particular, to support a visit by groups that do not have a joint goal (like a class) but instead have common bond, e.g., a family, or friends that visit together. The idea is that technology may help integrate their experience [18]. For that purpose we aim at developing the formal model of the collaborative activity of the visitors. The SharedActivity model could potentially support:

- *role-based presentation* by taking into consideration the role of the visitor in the group and provide him/her with information according to his/her role. For example, in the case of a family visiting a museum, the system will be able to explain to a parent about the objects that fascinate his/her child.
- *coordinate presentations* by updating the visitor about the objects which interest his/her group members. For instance, the system will be able to show a visitor what is of interest to her husband. On the other hand, if the visitor discovers an object that her husband has not yet seen, but may interest him, she will be able to send him relevant information.

- *generate group summaries* by sensing and interpreting what happens to each individual in the environment the system will be able to support members' interactions during and after the visit. For instance, it will be able to integrate presentations among the group's members and to support group summaries at the end of the visit.

- *helpful behavior capabilities* by reasoning 'what could help' the others in their visit. For example, if a team member detects an interesting activity in the museum that other members may like, but this area is too crowded, the system will be able to inform the others about this problem.

As described in chapter 4, we have tested our formal model on a limited simulation of a small group visiting a museum. We describe how we implement the above *helpful behavior capabilities* and *coordinate presentations*.

3.2 Overview of the Model

Figure 3.1 lists key components of mental states of members when they have a collaborative activity. First, cooperation implies the ability of the agents to identify themselves as members of a group (Item 1). Second, when the members engage in interaction, they may exchange information through verbal and nonverbal communication process. This maintains the group. The belief that members intend to be a part of the group gives the motivation to interact (Item 2). Third, each individual in the group is characterized by life histories, development patterns, needs, goals, and behavior patterns. These characterizes should be known by the other members during the collaborative activity and are represented in each of the member's profile. Thus, the members must have belief about the profile of others (Item 3). The profile may be given explicitly or implicitly (e.g., learning the profile by observation, overhearing, etc.). Fourth, dependence refers to the relation in which the utility that one member obtains from its own behaviors is affected at least partly by the activities of another party. Mutual dependence means that the utilities of all the parties are determined by their collective behavior [4] (Item 4).

Note that the key components are suitable for both treatment and task groups as these components do not consider the purpose for which the group is organized. Furthermore, in former models for supporting task groups (e.g., [6]), the agents hold the above mental states implicitly: First, they have joint intentions to achieve a joint goal which entails their beliefs about the members of the group and their intention to maintain the group during the performance of the task. In addition, the agents hold beliefs about intentions of other agents, their capabilities and their situations which can be considered as the profile. Since the members have a joint goal and their utility is attained by satisfying this goal together, their utilities are

To have a collaborative activity, a group of agents must have

1. mutual belief that all group's members are part of the group

2. mutual belief that all group's members have intention that the group be maintained

3. belief about the (partial) profile of other members

4. mutual dependence

Figure 3.1: Key components of collaborative activity.

determined by their collective behavior (i.e., they are mutually dependent). Thus, a task group is a special case of SharedActivity.

In the museum example visitors act in cooperation if they enjoy sharing their experience. In such a case, they must identify the group members who enjoy sharing the experience (clause 1 in Figure 3.1). Their intention to maintain a group and beliefs about the others' profiles entails exchanging information (clauses 2–3 in Figure 3.1). They believe that by sharing the experience they enhance the experience of each individual as well as of the group (clause 4 in Figure 3.1).

3.3 The Model Formulation

We use standard operators for intention and belief [6]. The operator Int.To$(A_i, \alpha, T_n, T_\alpha, C)$ represents A_i's intentions at time T_n to do an action α at time T_α in the context of C. Int.Th$(A_i, prop, T_n, T_{prop}, C)$ represents an agent A_i's intention at time T_n that a certain proposition $prop$ holds at time T_{prop} in the context of C. The potential intention operators, Pot.Int.To(...) and Pot.Int.Th(...), are used to represent the mental state when an agent considering to adopt an intention but has not deliberated about the interaction of the other intentions it holds. The operator Bel(A_i, f, T_f) represents that an agent A_i believes the statement expressed by formula f at the time T_f. Note that we abused the notation, and the formula f is not really the argument, but its name is $'f'$. MB(...) represents Mutual Belief. In addition, the operator Do(A_i, α, T_α) holds when A_i does action α over a time interval T_α.

The formal definition of SharedActivity (SA) is given in Figure 3.2 (clauses 1–4 are equivalent to the cases 1–4 of Figure 3.1). It specifies those conditions under which group $\mathcal{A}$ can be said to have a collaborative activity $\mathcal{C}$, at time $T_\mathcal{C}$. The activity $\mathcal{C}$ represents a set of actions which have been carried out by the group members during the collaborative activity. The collaborative activity may be associated with several properties such as constraints. Doing an action in the context

of the collaborative $\mathcal{C}$ must consider these constraints. For example, a parent and child visiting a museum consists of the actions of looking at objects, but may have the constraint that the parent and the child cannot move away from each other. We use the notation $\mathcal{P}$ to represent the profiles of the members and we denote by P_i^j the A_i's beliefs about A_j's profile. The operator $member(A_i, \mathcal{A})$ in the definition holds if A_i is a member of $\mathcal{A}$. In the fourth clause, the mutual dependence is specified by the utility of A_i from being a member of $\mathcal{A}$.

$SA(\mathcal{C}, \mathcal{A}, \mathcal{P}, T_{\mathcal{C}})$

1. $\mathcal{A}$ has MB that all members are part of $\mathcal{A}$:
MB$(\mathcal{A}, (\forall A_i \in \mathcal{A}) member(A_i, \mathcal{A}), T_{\mathcal{C}})$

2. $\mathcal{A}$ has MB that the group be maintained:
MB$(\mathcal{A}, (\forall A_i \in \mathcal{A}) Int.Th(A_i, member(A_i, \mathcal{A}), T_{\mathcal{C}}, T_{mem}, \mathcal{C}))$

3. Members of $\mathcal{A}$ have Bel about the profile:
$(\forall A_i \in \mathcal{A})Bel(A_i, (\forall A_j \in \mathcal{A})(\exists P_i^j \subseteq \mathcal{P}), T_{\mathcal{C}})$

4. $\mathcal{A}$ has MB that being a member obtains better utility:
MB$(\mathcal{A}, (\forall A_i \in \mathcal{A}) utility(A_i, member(A_i, \mathcal{A})) \geq$
$utility(A_i, \neg member(A_i, \mathcal{A})), T_{\mathcal{C}})$

Figure 3.2: Shared Activity

3.4 Axioms

In this section we present three axioms. These axioms further constrain the design of computer agents for shared activity. The axioms describe the behavior of the agents acting according to the model and guide programmers how to use the model.

As agent A_i may decide to adopt an intention to do an action α as a part of the SA. The agent's decision must take into consideration the benefit and the cost of performing α. Because of the mutual dependence between the members, when A_i performs the action α, $A_j \in \mathcal{A}$ may obtain reward from action α being taken. The mutual dependence between the agents is attained by the benefit function, $b_i^j(\alpha)$, and the cost function, $c_i^j(\alpha)$, where i denotes the agent A_i who is the performer, and j denotes the agent A_j.

A1. Cooperative act axiom. An agent A_i is cooperative when its activities do not contribute only to its own utility but also to the utilities of the other members. This axiom deals with two cases of a cooperative agent. The first case states that if

A_i believes that A_j is damaged from performing α by itself then A_i considers doing α. In the second case both agents, A_i and A_j, obtain benefit from performing and there are three options: First, A_i may consider doing α by itself. Second A_j adapts a potential intention that α will be done by A_j. Third, α will be performed by A_i and A_j jointly.

$(\forall \alpha \in \mathcal{C}, (\forall A_i, A_j \in \mathcal{A}), T_n)$
[Bel$(A_i, b_i^i(\alpha) - c_i^i(\alpha) > 0, T_n) \wedge$
Bel$(A_i, b_i^j(\alpha) - c_i^j(\alpha) > 0, T_n) \Rightarrow$
[Bel$(A_i, b_j^j(\alpha) - c_j^j(\alpha) \leq 0, T_n) \Rightarrow$
Pot.Int.To$(A_i, \alpha, T_n, T_\alpha, \mathcal{C})] \otimes$
[Bel$(A_i, b_j^j(\alpha) - c_j^j(\alpha) > 0, T_n) \Rightarrow$
(Pot.Int.To$(A_i, \alpha, T_n, T_\alpha, \mathcal{C}) \vee$
Pot.Int.Th$(A_i,$ Do$(A_j, \alpha, T_\alpha), T_n, T_\alpha, \mathcal{C}) \vee$
Pot.Int.Th$(A_i,$ Do$(\{A_i, A_j\}, \alpha, T_\alpha), T_n, T_\alpha, \mathcal{C}))]]$

Note that the above axiom may lead to the formation of a task group which is handled by previous models. Such an opportunity for forming a task group occurs when A_i and A_j mutually believe that they have utility from performing α by themselves and both of them adopt intentions as given in the third option of the second case. However, we leave the discussion of how agents can recognize and take advantage of such opportunities for future work.

In the museum example, coordinated presentations are a type of cooperative activity. If A_i looks at an object which is of interest to both A_i and A_j, both of them obtain a benefit, as they may share their experience. In the case that A_j is far from the object and her cost to arrive to the object is too high then A_i may look at the object for both of them and notify A_j about information which is interesting for A_j.

A2. Helpful-behavior act axiom. An agent A_i may help another member A_j, even if A_i does not obtain any benefit from the performance of α. The following axiom states that an agent A_i will consider taking action α which may decrease its utility, if it believes that its cost is bounded by some lower bound (LB). Also, A_i believes that, by performing α, A_j obtains significant benefit (i.e, A_j's greater from f_1 on A_i Utility.) In addition, A_i believes that its loss from performing α is significantly smaller than A_j's loss when A_j performs α.

$(\forall \alpha \in \mathcal{C}, (\forall A_i, A_j \in \mathcal{A}), T_n)$
[Bel$(A_i, LB < b_i^i(\alpha) - c_i^i(\alpha) \leq 0, T_n) \wedge$
Bel$(A_i, b_i^j(\alpha) - c_i^j(\alpha) > f_1(b_i^i(\alpha) - c_i^i(\alpha)), T_n) \wedge$
Bel$(A_i, b_j^j(\alpha) - c_j^j(\alpha) < b_i^i(\alpha) - c_i^i(\alpha) + \epsilon, T_n) \Rightarrow$
Pot.Int.To$(A_i, \alpha, T_n, T_\alpha, \mathcal{C})$]

The role-based presentation, in the museum example, is a type of helpful-behavior act. In such a case, A_i looks at objects which are of no interest to her but

interest A_j, she does it in order to increase the experience of A_i.

A3. Selfish act axiom. The following axiom states that an agent A_i will consider taking an action α when A_i believes that it obtains some benefit from α's performance but A_j does not obtain any benefit. Also, as a member of the group, A_i cares that A_j will not be damaged from the performance of α (i.e., the loss of A_j is greater from f_3 on A_i Utility)

$(\forall \alpha \in \mathcal{C}, (\forall A_i, A_j \in \mathcal{A}), T_n)$

[Bel$(A_i, b_i^i(\alpha) - c_i^i(\alpha) > 0, T_n) \wedge$

Bel$(A_i, f_3(b_i^i(\alpha) - c_i^i(\alpha)) < b_i^j(\alpha) - c_i^j(\alpha) \leqslant 0, T_n) \Rightarrow$

Pot.Int.To$(A_i, \alpha, T_n, T_\alpha, \mathcal{C})$]

In the museum domain, looking on an object which is not of interest for other members in the group is a selfish act. The values of ϵ_k $(k = 1, 2, 3)$ and LB in the above axioms are influenced by several parameters [4] such as moral principals, the relationship between the members, the members reputation, etc.

Chapter 4

Experimental Design and Analysis

To explore the behavior of the SharedActivities model we developed a simulated museum test-bed, in which we could vary different factors influencing the behavior of the agents. The museum was represented by a weighted connected graph. Each vertex in the graph denoted a picture. Vertices were organized into small cliques. Each clique simulated a room. Common vertices between cliques simulated doorways. In the experiments below, we used museums with 10 rooms, and 8 pictures in each room. Each agent had a profile that indicated the agent's estimation of the value of each picture.

For each profile, the museum was organized so that each room contained pictures with similar estimated values; since the variance between the values of the pictures was defined by a small value, the simulated museum had rooms with certain topics for each room. Also, each room had at least one neighbor's room with a similar estimation, simulating a connection between the topics of the two rooms.

We simulated agents which could tour the graph. The duration of the tour was limited to a fixed amount of time. Each agent was able to perform four actions: (a) "looking at a picture" denoted Look; (b) "going one step in the museum" denoted Move; (c) "sending a broadcast message" denoted SendMsgs; and (d) "exchanging information" denoted Exchang. The goal of each agent was to maximize its total utility which was composed from benefit and cost. The actions Look and Exchang yielded some benefit and the actions Move and SendMsgs yielded a cost. The agents were obligated to get into the museum and to get out.

Prior to the tour, the agent estimated the rooms that maximize its total utility (with some level of uncertainty which we vary in the experiments) and based on a heuristic function it built a plan of tour. During the tour the agent could receive more accurate information about rooms (from other agents) and change its original plan based on this information. A change happens if the agent receives information on at least half of the pictures in the room from another agent saying that there is a variance between the estimations of the pictures. If the grade of

the room changes, it might change the original room tour plan. When an agent arrives in a new room, it decides which series of actions will maximizes its utility according to a greedy heuristic function.

We simulated five types of groups ranging from fully cooperative to independent. In all the groups, members were acquainted with their profiles. Each member could know its location within the graph. The heuristic function for estimating the value of a picture to a profile, could be different between the groups; the difference between the heuristic functions represents the different evaluations of the pictures. Each group had a different commitment level, ranging from a high to a low level.

1. Task group denoted TA, acted as a group with a joint goal of maximizing the total group benefit, used the Cooperative act axiom because the agents are cooperating all the time. Each member visited pictures that maximized the utility of the group. Each agent was committed to notify the others about its location so they would calculate the group benefit from these pictures. The rank of the rooms is according to the utility of the group.

2. Treatment group acted according to the axioms of the SharedActivity model and used all of them. In this case, each member tried to maximize its own benefit, and cooperated with others according to its individual decisions which were based on its situation and its beliefs about the others. Each member could know the room location of the other members within the museum by getting this information from the others. Cooperative activities took the form of visiting pictures that were beneficial for other members and also to themselves. Helpful activities took the form of visiting pictures that would help for other members. Sharing information reduced the uncertainty of the other members on the utility of visiting rooms, and thus affected their planned paths. The agents shared all the relevant information at the end of the tour. We examined two levels of cooperation:

   (a) Strong treatment group (more helpful) denoted $S-TR$. Each member could know also which picture no one has yet seen. Helpful activities took the form of visiting pictures that would complete information on the room for other members, so they will be able to re-evaluate. Another helpful activity was to visit pictures that no one has seen that could contribute to other agents.

   (b) Weak treatment group (less helpful) denoted $W-TR$. Helpful activities took the form of visiting pictures that would complete the information on more than half of the pictures in the room for other members, so they would be able to re-evaluate.

3. Group of purely individuals agents denoted as $G - IN$. The member did not hold the mental states of the SharedActivity model, but they cooperated at the end of the tour. The cooperation was expressed by sharing all the relevant information at the end of the tour. The information's handout was egoistic and only for retrieving knowledge from the other. Beside that, they always maximized individual utility.

4. Group of purely solitary agents denoted SO. The members did not hold the mental states of the SharedActivity model. They weren't aware of the other agents. Therefore they could not cooperate in maximizing their own utilities. They always maximized individual utility.

We ran extensive experiments to measure the difference between the groups using this environment[1], varying:

1. The agents' number (AgentN).

2. The duration of the agents' stay in the museum (Time).

3. The level of uncertainty (UncerL) regarding the expected benefit of visiting pictures (given as a range around the actual interest level).

4. The similarity measure between the member profile.

5. Cost of sending one broadcast message $c_i^i(\text{SendMsgs})$ denoted C_m.

6. Cost of going one step $c_i^i(\text{Move})$ denoted C_s.

We measured the total utility which was composed of three elements:

1. Independent benefit - $b_i^i(\text{Look})$ A_i's benefit from actions that it does by itself. The agent obtains a benefit when it arrives at a picture. Each agent has a profile that matches its interests with pictures. Looking at a picture yields a utility in the interval [-3, 7]. For each time-unit that the agent stayed at the picture, the profit decreased (i.e. if the value of the picture is 6 and it stayed there two time-units, at the first time-unit it will get 6, at the second it will get 5 and the next time it comes to this picture, it will get 4). Prior to arrival at the picture, the agent could only estimate the utility that could be generated by the visit, with some uncertainty (which we vary in the experiments).

[1]The total number of combinations we tested exceeded 30,000. For each such combination, we generated 36 trials, for a total of just over a million runs. We report on a small subset of these results, highlighting key lessons below.

2. Dependent benefit - b_i^j(Exchang) A_i's benefit from actions that A_j did for it. At the end of the tour the agents exchanged information about the pictures that interested the other agents in the group. If some agent A_i looked on a picture that interested an agent A_j then when A_i informed A_j of this picture, A_j obtained some benefit from this information (in our simulation, it was 1/3 of the original benefit value $b_i^j(\text{Look}) = 1/3 \times b_j^j(\text{Look})$). If more than one agent informed A_j about this picture it increased the benefit only by 1 point (but it was bounded up to four agents).

3. Cost - We distinguished between two types of cost for analyzing the group's behavior:

   (a) Cost of all messages - the number of messages multiplied by C_m (i.e., $\sum c_i^i(\text{SendMsgs})$).

   (b) Cost of all steps - the distance that passed multiplied by C_s (i.e., $\sum c_i^i(\text{Move})$).

Each element directly influence On the other elements. Therefor we present their combination only.

4.1 Influence of Time

The average number of the changes that were done in the plan of the agents during their tour as a function of time duration is given in Figure 4.1. We measured the average number of the changes that were done in the plan of the S-TR group only. In lower time duration, we found that the agents did not have enough time to implement changes in their plans. On the other hand, at higher time duration, they did not have reasons to make changes because they had enough time to tour all over the museum. We will report on experiments with duration of 360 (Time = 360) units since in these settings the cooperation significance.

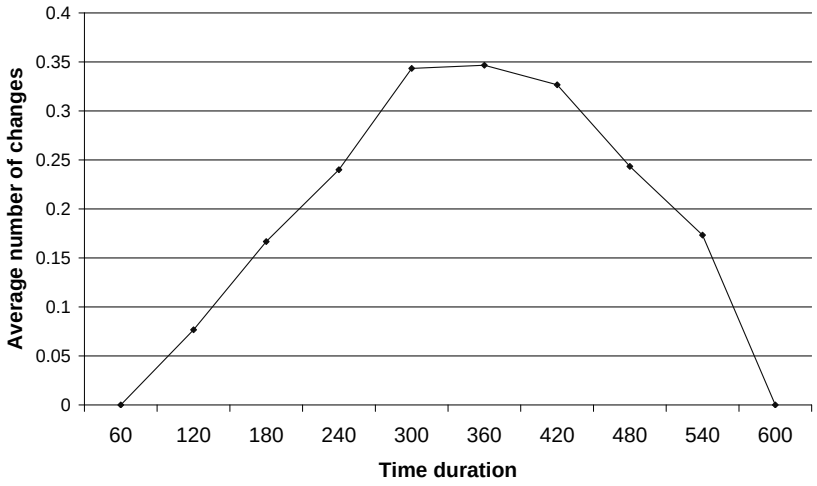

Figure 4.1: The average number of the changes as a function of time duration($Time = [60, 600], C_s = 1, C_m = 0.5, AgentN = 6, UncerL = 1$)

4.2 Influence of Message Cost

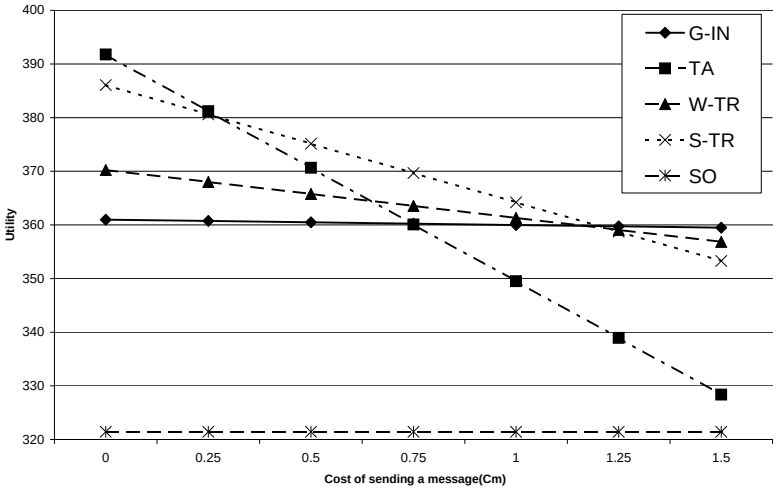

Figure 4.2: The average utility as a function of the messages' cost($Time = 360, C_s = 1, C_m = [0, 1.5], AgentN = 6, UncerL = 1$)

The benefit obtained by the agents as a function of a message cost is given in Figure 4.2. When the message's cost is very low, it is better to be in the TA or in the S-TR since they use messages heavily. As the messages' cost rises it is better to be selfish, since selfish agents hardly use messages.

When the message's cost value is 0.5, the differences between all the groups are significant. Moreover, 0.5 is the value in which the differences are maximized. Therefore, in the rest of the experiment, we will report on results where the message cost value is 0.5.

The SO group do not use messages at all, therefore the message's cost do not influence them at all. The benefit of the SO group is very low, since the agents don't replace relevant information at the end of the tour, and therefore we have not show the SO utility in the others figures.

4.3 Influence of Similarity and Uncertainty

We also checked the influence of different rates of similarity between the profile of the group's member under the same environment setting. We compared between three similarity setting:

1. When the profile of all the agents was similar.
2. When the profile of all the agents was very different.
3. When the profile of all the agents splits uniformly.

Similar profile means that the agent will get the same benefit from each picture. Different profile means that the agent will get benefit with difference of 4 in average from each picture. Splits uniformly means that the benefit of each agent from a picture splits uniformly.

We checked the influence of different setting of the profile similarity under different levels of uncertainty.

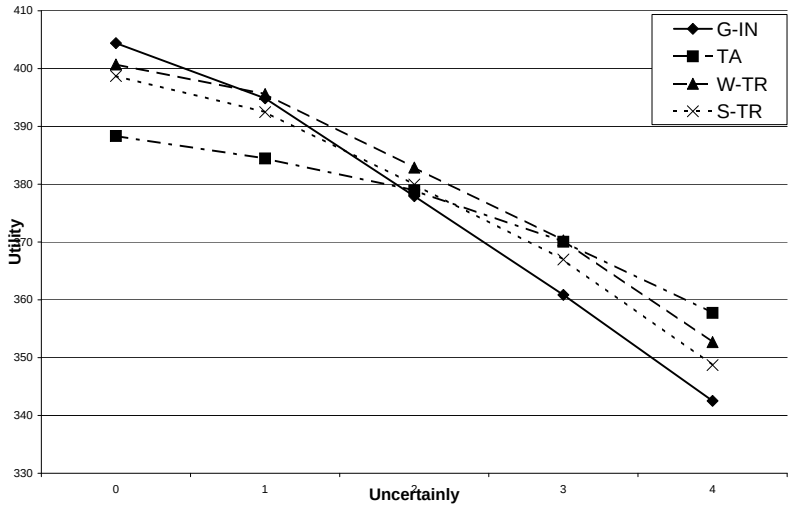

Figure 4.3: The average utility of groups, for agents that were similar, as a function of the uncertainty($Time = 360, C_s = 1, C_m = 0.5, AgentN = 6, UncerL = [0, 5]$)

When we examine the figures which presents the influence of the uncertainty (Figures 4.3 – 4.5) we can see that as the uncertainty rises the agents earn less.

When the agents cooperate they reduce the uncertainty, but this depends on profile similarity.

In Figure 4.3, the benefit that similar agents get is higher compared to non-similar agents in the other figures. The reason is, all of them go to pictures that are interesting for all of them and therefore increase the benefit of all the agents. In contrast, in Figure 4.4, the benefit that very different agents get is lower compared to all other figures.

In Figure 4.3 when the agents have similar profiles it appears that when the uncertainty is low, it is better to be in G-IN. When the uncertainty is high it is more profitable for an agent to be in the TA group. When they cooperate they reduce the uncertainty but because they are similar, they go to similar rooms so the overall uncertainty reduction is limited (there are a lot of places nobody goes, so nobody reduces the uncertainty of those places). Therefore the cooperation is limited. In this case, the only essential difference between W-TR and S-TR is that agents in S-TR send more messages but they do not go to see pictures that are good for other agents because those picture that are good for them are good for all the other agents, due to the similarity between the agents. Therefore, S-TR groups gain less from the cooperation relative to the W-TR groups. The TA groups do not act according to the rule of 'helpful' but according to the group's profit, so they pass all the groups as the uncertainty rises. On the other hand, the G-IN groups do not cooperate so as the uncertainty rises, they earn less relative to the cooperators.

In Figure 4.4 when the agents have different profiles it is more profitable for the agents to be an S-TR group. The reason for the advantage of the S-TR is the difficulty to find a profitable picture that will be profitable for the other agents as well. The S-TR agents do not look only for those pictures that benefit themselves, they also look for pictures that may only benefit other group members. Because of the large differences between the agents, the uncertainty does not influence the relationship between the cooperative groups in any significant way. This is because all of the agents go to pictures that do not interest the others and therefore barely decrease the uncertainty for the others. Regardless, there is some mutual help and therefore as the uncertainty rises the G-IN earn less, to the cooperative groups.

In Figure 4.5 when the agents have profiles that are split uniformly, it is more profitable for agents under these circumstances to be an S-TR group (paired t-test, $p < 0.0003$). Where the price of each step is 1 and the price of each message is 0.5, medium investment in a group is sufficient to get more profit then the other groups. TA agents invest to much in the group while W-TR agents do not invest enough. More specifically, as the uncertainty decreases, the W-TR benefit less relative to other groups (the help is less significant). In the TA the trend is that there is a very high overhead regardless of the uncertainty. Therefore, when the uncertainty is low, there is a high loss for little gain. However, at high uncertainties, the cost

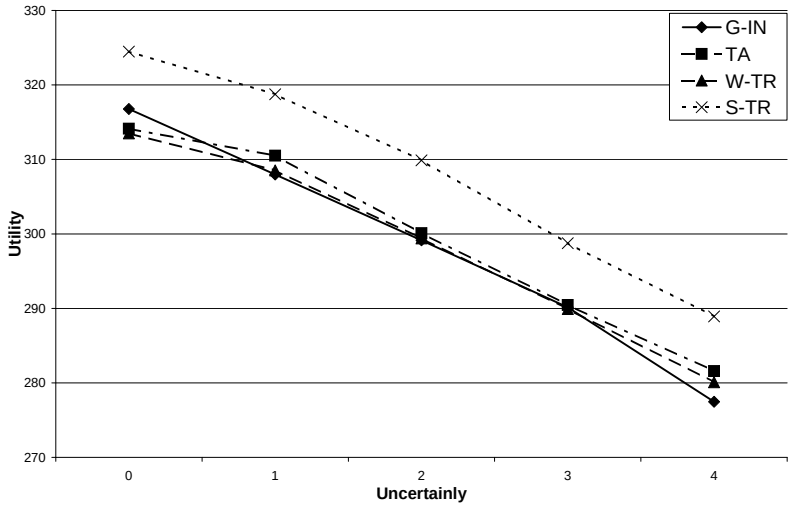

Figure 4.4: The average utility of groups, for agents that were very different, as a function of the uncertainty($Time = 360, C_s = 1, C_m = 0.5, AgentN = 6, UncerL = [0, 5]$)

does rise significantly, and therefore there is a large gain; it is possible to see the beneficial results of the cooperation by the increase in the gap relative to the G-IN group that does not cooperate. When the uncertainty is high there is no difference in the trend between the groups because all of them manage to help at high uncertainties.

When the agents have similar or different profiles we measure two indexes that are preferable to the agents that have profiles that split uniformly:

1. There is at least 40% less helpful behavior.

2. Information's cooperation cause between 30% to 50% less changes in the tour.

Situations with more consumes to cooperation are more interesting and therefore in the rest of the report we concentrate on agents with profiles that are split uniformly.

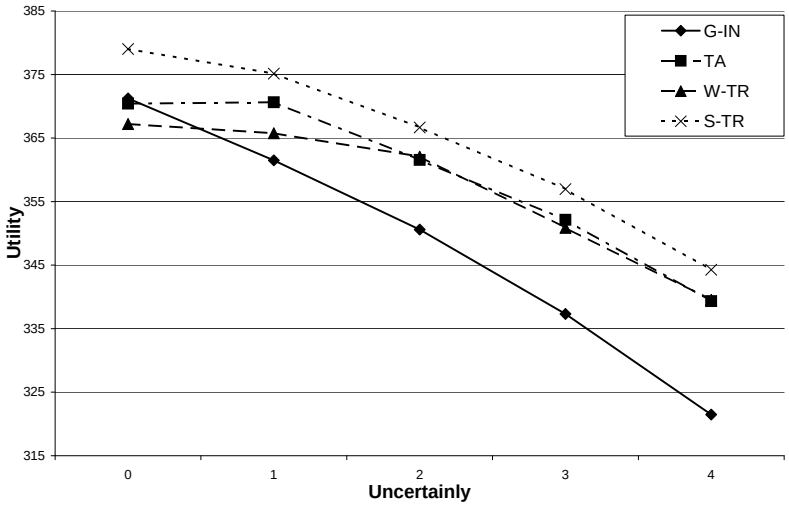

Figure 4.5: The average utility of groups, for agents that their profile splits uniformly, as a function of the uncertainty($Time = 360, C_s = 1, C_m = 0.5, AgentN = 6, UncerL = [0, 5]$)

4.4 Influence of Group Size

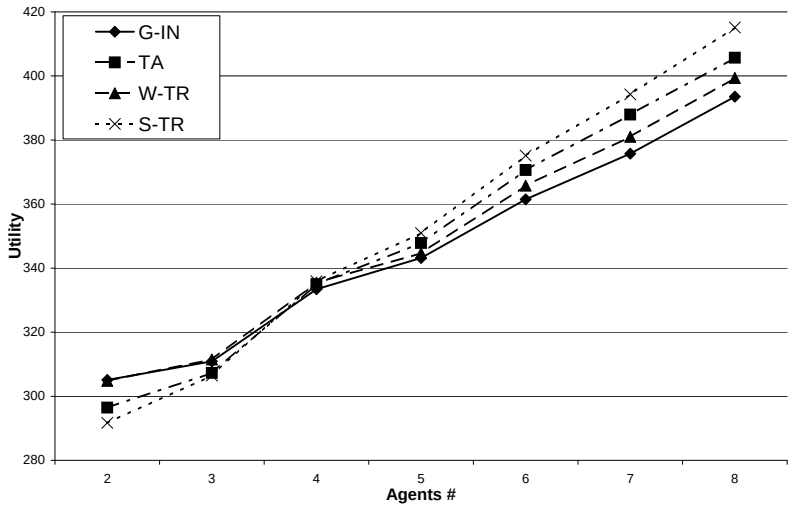

Figure 4.6: The average utility as a function of the agents' number ($Time = 360, C_s = 1, C_m = 0.5, AgentN = [2, 8], UncerL = 1$)

In Figure 4.6 we turned to study the effect of the number of the agents in the group on the benefits gained. We anticipated that increasing the number of the agents in the groups will increase the benefit, since they get more help with reduction of uncertainty and more information, essentially an increase in the Depending benefit, and this indeed did happened. When the groups are small, the accompanying addition from the cooperative is high relatively to a profit and therefore groups that are less cooperative earn more(paired t-test, $p < 0.0001$). The groups TA and S-TR are more cooperative and therefore as the amount of agents rise, the gap between them and the other groups is reduced. The W-TR agent's investment in the group is sufficient in order to get more profit than the other groups. TA agents invest to much in the group while W-TR do not invest enough and that is why it is more profitable for an agent under these circumstances to be in a S-TR group.

4.5 Influence on Maximum and Minimum Benefit

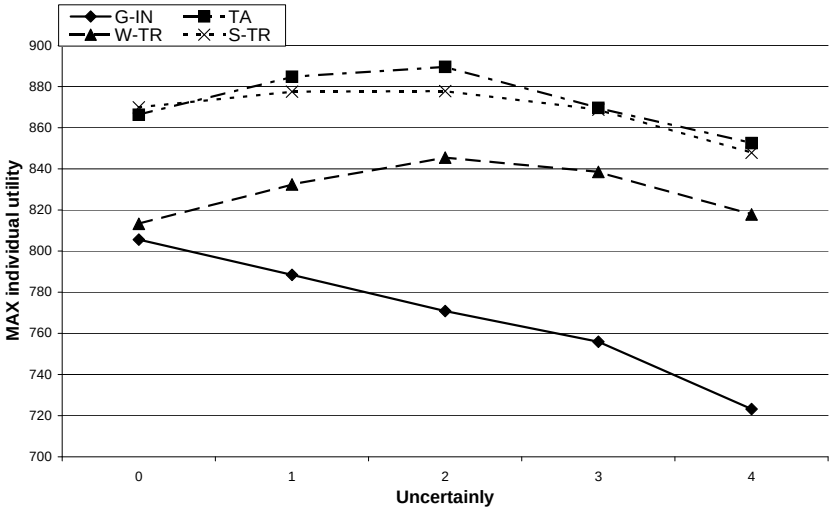

Figure 4.7: The average utility of the maximal utility agent as a function of the uncertainty($Time = 360, C_s = 1, C_m = 1, AgentN = 6, UncerL = [0, 5]$)

Figure 4.7 and Figure 4.8 show the influence of the groups on the average benefit of the agents who earned the maximum (or the minimum, respectively) utility in each museum. The value of each point in the graphs is equal to the average of all the agents who get the maximum (or the minimum) utility in each museum, over 36 trial.

In Figure 4.7 we assumed that as the uncertainty increases, the utility decreases. To our surprise, we discovered that it did not happen when the agents cooperated. And indeed as the uncertainty becomes larger, more cooperation is necessary (i.e., the sharing of the information between the group members help to increase the utility). As a result, the graph in Figure 4.7 is increased when the uncertainty increased. Yet, the utility start to decrease when the uncertainty becomes too high (above 2). The reason for this fact is that when the uncertainty is smaller than 2 it is easier to decide about the activities that may help to the other members in the group and more help is given, in particular to the member that obtains the highest utility in the group (as there are more pictures that contribute to its utility). As a consequence, when the uncertainty is too high it is hard to find out the activities that help to others. In more cooperative groups, the agent that obtains the highest utility in each group, earns more.

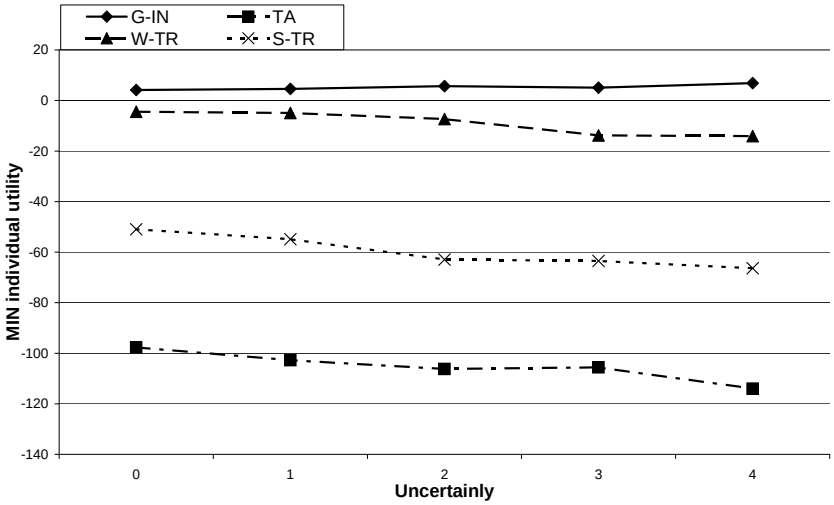

Figure 4.8: The average utility of the minimal utility agent as a function of the uncertainty($Time = 360, C_s = 1, C_m = 1, AgentN = 6, UncerL = [0, 5]$)

In Figure 4.8 we can see that the only group in which poor agents never have negative utility is the individual's group. In the rest of the groups, self-sacrifice is required sometimes, and as the uncertainty increases the utility decreases. The more they cooperate, they sacrifice more. However, as we saw in previous figures, in the long run it is worthwhile.

4.6 Influence of Steps' Cost

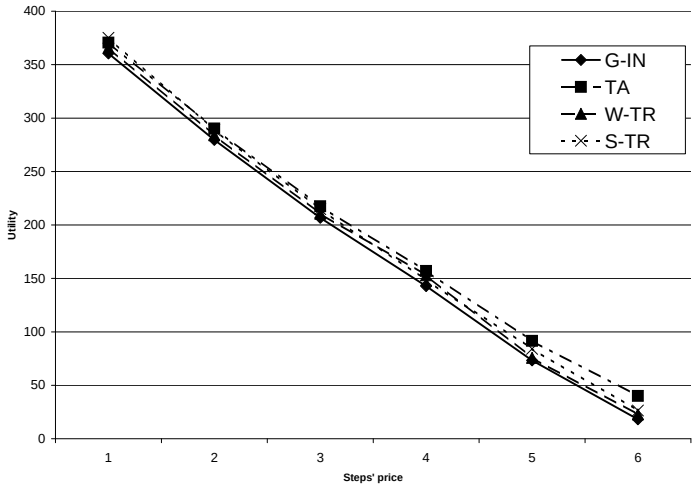

Figure 4.9: The average utility of the last as a function of the uncertainty($Time = 360, C_s = 1, C_m = 0.5, AgentN = 6, UncerL = [0, 5]$)

When we checked the influence of the steps' cost we discovered (Figure 4.9) that when the cost value is more then 2 it is significantly better to be a TA group (paired t-test, $p < 0.001$). When the steps' prices were low (less then 2) the TA group got less than the S-TR group.When the steps' price was high (more then 2), assistance from others becomes much more valuable. The TA agents were the only agents that were committed to such assistance, and therefore earned more.

4.7 Discussion and Summary

The results represent only the group's behavior in our simulation but it is possible to extrapolate from these conclusions to other environments. It is better to be in the selfish group that shares information at the end of the tour in these situations:

1. There are fewer agents.
2. Small uncertainty.
3. The messages' price is high.

It is better to act according to our model in these situations:

1. There are more agents.
2. There is uncertainty.
3. The messages' price and the steps' price are not high.

It is better to be in a task group only if the step's price is very high; indeed, it is more expensive to cooperate in this case, but a commitment to the group is the best way to deal with this situation. This scenario simulates a situation where we must be obligated to cooperate if we want to succeed, and therefore this is the case where we will act as a task group.

Most often its better to be at a medium level of cooperation, so as to not pay too much for the group and to not earn too little.

Chapter 5

Conclusions

We presented a model of collaborative activity for supporting cooperation between group members consisting of humans and computer systems. Based on studies from psychology, the model suggests key components and mental states for agents who act in cooperation. In contrast to former models it deals with both treatment and task groups and allows different levels of cooperation.

We investigated the behavior of the model in a simulation environment and compared between benefits attained by being members in treatment and task group as well as acting as being a selfish motivate. Results show that the benefit is influenced from different parameter settings, in case of sufficient resources to achieve the individual goals the treatment group attains the best benefit. However, when the resources are limited it is better to act as a task group.

As future work, we suggest to develop a formal model of collaborative activity for a dynamic group. and to find a formula that will help an agents to decide the level of cooperation that will provide them the best benefit in the future. In addition we want to find how agents can recognize and take advantage of situation where A_i and A_j mutually believe that they have utility from performing α by themselves and both of them adopt intentions to perform it.

Part II

Iterative Search in Cooperative Closed Groups

Chapter 6

Introduction

To be in shared activity in our era, there is no obligation to be in close proximity. Shared activity can be through cellular devices. In the cellular era it has double meaning. First, relationship through communication, and second helping with the assistance of sending objects like MMS (mobile multimedia messaging). For example, Battarbee [1] defines ”Co-experience” as the user experience, which is created in social interaction. She describes shared activity of people that shared MMS.

In a virtual album application in a closed group, the group members share pictures stored in their cellular phones. Often, the group members would like to look at pictures that are not stored on his own cellular phone. The group members are cooperative, know each other, willing to share their pictures and to spend resources in helping others obtaining pictures even though no payment is given in exchange for such help. When a user is looking for a picture, he or she will specify keywords that characterize the required picture, locate the picture and download it to their own cellular phone.

This is an example of an iterative search in cooperative closed groups. The agent receives from the user a request to search for a resource(picture) characterized using a few keywords. The user's agent that controls the virtual album application should find and display the requested picture as quickly as possible. The Album agent should obtain the picture at the shortest time and the lowest price. Additionally, if the agent is approached by another agent with a request for a picture, the agent will make an effort to respond to such a request.

This virtual album application is an example of a shared activity [7]. The group has a mutual interest to keep the group maintained. Each member has beliefs about the profiles of the others. All the members have a mutual belief that being a member leads to higher utility, and are willing to help each other. However, there are limits on their willingness and capabilities to spend resources when helping others.

Thus, it is important to provide protocols and strategies that are efficient in distributing the load among the members in a fair manner. In this application, there are two sources for such a load: First, in responding to search queries, and second, in actually transmitting the pictures. Load of the first type is usually incurred in locally examining the picture database for pictures matching the query. Load of the second type is incurred because of the transmission delay and transmission costs (in some payment models).

Taking advantage of the fact that we are addressing a closed-group, we can—in principle—use a full-graph topology to conduct searches, while limiting the number of messages, and reducing the waiting time for the user. One naive way of doing this would be to conduct brute-force search. Here, the searching agent contacts all other agents in order to locate a picture. However, such a search is too costly in bandwidth use. Seemingly it finds who has the object in the faster way but it yields a high load for the group. In particular, many unnecessary messages are exchanged. Furthermore, if it waits for the answer from all group members to identify the one that is not loaded, than this type of search may also *increase* the time of obtaining the picture. Thus a key issue is the uncertainty of the other's state and willingness to respond.

Our objective would therefore be to (1) decrease the number of query messages that are sent to any one individual agent; and (2) reduce the number of picture transmissions from any single agent. This, while (3) keeping the time it takes to acquire a picture to a minimum.

Chapter 7

Related Work

In an effort to curb uncertainty about the load on another agent, one may theoretically attempt to use any of the different methods for computing optimal policies in the context of uncertainty. MDP (Markov Decision Processes) is a method to model stochastically-changing environments and for identifying optimal policies for actions. However, it does not deal with partial observation which is the main challenge in our problem.

The POMDP (Partially-Observable Markov Decision Processes) model accounts for partial observations. In particular, Communication-Decentralized-POMDP [5] should be considered. However, while the virtual album group members have some joint goal of maintaining the group, each of them also has personal goals that usually have higher priority than the joint one. So, adjusting Communication-Decentralized-POMDP to our application is problematic. Furthermore, our actions' results are without uncertainty (we assume reliable communication). The system's uncertainty is derived from the partial observation, not knowing which of the pictures are stored by other agents nor the actions they have taken. Moreover, we do not have the reward function of the other agents but an estimation of their reward. Finally, because of the huge number of states in our application, the complexity of finding an optimal policy, even if we will relax some of the uncertainty, is too high [2] and we will not be able to calculate the optimal policy in advance given the limitation of CPU and memory of the cellular devices.

Due to the missing knowledge on the agents' rewards and settings, one could consider applying classical reinforcement learning (RL) model which deals with learning of value-functions or searching for efficient policies [21]. However, the agent's state is changing over time and the situation is too complex as to apply reinforcement learning directly. However, we do apply a method that tries to estimate the load and the willingness to help the other agents and update it repeatedly based on the responses of the other agents, as well as their explicit messages. If users will be provided with a feature to determine their level of helpfulness,

reinforcement learning may be applied to learn this value.

Classification of the data that each agent holds will help to reduce the size and the amount of the transferred messages. Müller et al [15] suggest to determine a certain number of clusters for the whole data collection. Under this assumption each agent distributes the information about the number of data in each cluster to all other agents. In this way the agents spread their profiles with less messages. Noervaag et al [17] improve this attitude and suggest taxonomy caching approach. According to their approach they assume that files are classified as belonging to one or more categories of a taxonomy. While focusing especially on the feature of a taxonomy, which is when an object belongs to a category, it also belongs to its ancestors in the taxonomy tree. Agents can send or ask for data according to these attitudes.

A different approach to modeling our problem is as a form of routing, using routing protocols for sharing data in wireless networks [12, 23]. There are two key differences between routing and our model: First, they deal with the network layer and we consider the application layer. Second, the network topology is different. The routing- works consider networks in which devices can communicate only with their direct neighboring devices. We, on the other hand, take advantage of the closed group and allow communication between all the devices.

Many search protocols for distributed systems have been developed for P2P systems. It is important to clarify the differences between our environment and classical P2P environments:

Group openness. We consider closed, and typically small, groups. All agents know all others. P2P systems, on the other hand, are open, and thus agents do not know all of their group members. Moreover, P2P systems typically deal with thousands of peers.

Network structure. Our agents are deployed on a full graph topology, and each P2P agent has a limited number of neighbors.

there are two different kinds of P2P networks, structured and unstructured. We focus on unstructured systems [24] for the following reasons:

1. The limitation of the memory.

2. The cost of messages that are needed for initializing the system (we consider relatively small groups).

3. The number of needed messages for updating the system after each change.

A popular approach for P2P unstructured systems is random walk [14]. In our case, we are using a more sophisticated choice algorithm which includes some

randomization. Nevertheless, random-walk search algorithms is effective for localization of resources, and their subsequent sharing. Thus such algorithms might be useful here as well. Other issues that concerned the P2P system is fairness (dealing with free-riding) [3]. Our proposed method provides a high level of Fairness; however, we are looking at it from a different perspective. For example, if two agents have the same picture and one of them has another unique picture, we will prefer to ask for the joint picture from the one who does not have the unique picture.

Random-walk procedures take a different take on searching. We can still take advantage of the fact that we have access to all agents, but instead of approaching all agents (as in brute-force search), we select only k agents and send them the query. We then wait for their responses: If one or more has the picture, we randomly select an agent to request the picture from; otherwise, we approach a new set of k randomly-picked agents.

The randomization procedure is meant to even the load on the agents. In the limit, there would be a uniform distribution of queries and transmission requests, which would balance the load on the agents. However, it may be slow since it does not take the load of the other agents into consideration. It may thus wait for a response, or wait for picture downloading, from an overloaded agent. Indeed, as we shall show in the experiments, random-walk searching has proven less effective than the algorithms we develop in the next section.

Chapter 8

Basic Search Algorithm

We developed a solution based on distributed decision-making within the SharedActivity model. We follow the general skeletal procedure for random-walk searching, and acquisition of the pictures: Query k agents, choose among successful responses those to contact for the picture. However, in our procedures, the decision of an agent on whom to contact is based on estimated load of others and an estimation of others willingness to help it (e.g., based on their own load). Similarly, an agent decides how to react according to its current load and according to its willingness to help the requester. The proposed algorithm supports limited-resource platforms, is fast, and fair.

We now describe this algorithm. Let us begin by defining the entities of the system. A closed group G is a set of n agents $G = \{A_1 \dots A_n\}$, where all agents mutually believe in G, and can contact any member of G. An agent $A_i \in G$ is a tuple $\langle O, ld, WT, EOWT \rangle$, where O is a set of m objects' identifiers $O = \{o_1 ... o_m\}$, ld is the normalized load of the agent depends on the communication load, battery and other features. WT is a set of n willingness thresholds $WT = \{wt_{i,1} ... wt_{i,n}\}$, where $wt_{i,j}$ is the willingness threshold of agent A_i in respect to agent A_j. $EOWT$ is a set of n thresholds $EOWT = \{eowt_{1,i} ... eowt_{n,i}\}$, where $eowt_{j,i}$ is the estimation of agent A_i about the willingness threshold of A_j in respect to agent A_i.

The willingness value and the estimation about other willingness value is initialized to $initial$ and during runtime is affected by the number of times A_i assists and is assisted by A_j.

In order to treat the requests of the users to search for objects, the agent should provide the value of specific parameters, which define the search. A search S is a tuple $\langle obj, maxt, maxc, PR, end \rangle$, where obj is the requested object's identifier, $maxt$ is the maximum run-time that the agent allows the search application to run, $maxc$ is the maximum number of messages that the agent allows the search application to send, PR is a set of n probabilities $PR = \{pr_1 \dots pr_n\}$. This set

represents the probabilities that other agents in the group (agents $A_1 \ldots A_n$) have the object, where pr_i represents the searching agent's view of the likelihood that agent A_i has the object obj. end is a boolean variable which is $true$ if the search is completed, and $false$ if the search is not completed. A search is completed if one or more of the following conditions are satisfied:

- The object is found.
- $maxt$ is reachd.
- $maxc$ messages were sent.
- Provably, no agent in G has the object.

The search agents executes two processes in parallel:

A Request process that treats the requests of its user to search for objects.

A Response process that responds to requests of other agents to receive objects from its user (or decides not to).

8.1 The Request Process

In the request process the agent assists the user to find objects in the group. This process includes two sub-processes in parallel. The first sub-process is responsible for requesting the object from other agents (described in Algorithm 1, $REQUEST\_OBJECT$). The other sub-process is responsible to collect the responses from the agents (described in Algorithm 2, $ANALYZE\_RESPONSE$).

The search agent estimates the likelihood that the other agents in the group have the object, their current load and their willingness to send the object. It first queries agents with a high likelihood of having the object, high likelihood of willingness and a low estimated load. If there is not enough information on these parameters then it asks agents with unknown probability to send it information about their current load, willingness towards the agent, and whether they have the object. This information enables the search agent, in the next iteration, to request the object from agents with high likelihood to send it.

This procedure is presented in Algorithm 1, $REQUEST\_OBJECT$. The main loop in the algorithm continues to run unless the requested object is found or alternatively it is not found through all the agents or it is terminated by the max time ($maxt$) or max communication ($maxc$) thresholds (line 3). In line 4 the agent estimates the maximum number of agents to ask for the object based on the

max time ($maxt$) and max communication ($maxc$) thresholds received from the user. The function return_max_agents_to_request decides on how many agents can be approached, at a maximum, to request an object once it is found. This would be set based on the preferences of the user for the thresholds $maxt$ and $maxc$, and the possibility of utilizing multiple sources. In the experiments, it always returns 1. This value is to be changed if multiple download sources are possible.

In line 6 the agent calculates which agents have the object with probability higher than the constant K, estimated high willingness, and estimated low load. This is done in the function return_high_probability_agents, which receives the probability set ($PR \in S$), the willingness estimation set ($EOWT \in A_i$), the maximum time threshold ($maxt$), the maximum communication threshold ($maxc$), and the constant K (which is application-dependent). In the experiments, We arbitrarily set the probabilities of PR to their extreme values (0 or 1), to evaluate the boundaries of the technique, and $maxt$ and $maxc$ were ignored, as their use is user-dependent.

Lines 9–12 begin the query process of the high probability agents. If the agent does not have information about agents with probability higher than constant K or its estimate of their willingness is too low or not up-to-date, it will continue to the exploration phase (line 14–17). In the exploration phase it will select a subset of the agents (the function return_unknown_probability_agents returns the four agents with unknown probability but with the highest willingness) among the agents with a probability lower than constant K, to send it if they have the object and are willing to send it. The agent updates its estimate of the willingness of the agents in the team to help it in lines 10 and 16. If it requests an object (line 10), then the estimated willingness is reduced by C. If it sends a message in line 16, then the estimated willingness is reduced by c. These constants will need fine-tuning depending on the application context.

With every cycle of the Algorithm REQUEST_OBJECT (Algorithm 1), the agent updates its estimates of the load of the other agents. With time, the estimates decrease, to allow for changes occurring with the passage of time.

The second sub-process that the request process executes, collects the responses of the agents in the group to the query. It analyzes these responses in order to update the probability that they have the object and its estimates of their willingness to send it. This sub-process is responsible for terminating the search once the object is found. This process is described in Algorithm 2). Note that typically, the responses of the agents are either 0 (does not have the object) or 1 (has the object). Even if the responding agent does not have the object, it must still respond, so that the requester knows not to ask it in the future (as opposed to an agent that may have been too overloaded to respond).

We now turn to the PROCESS_ANSWER algorithm (Algorithm 2). The agent analyzes the received message. First it associates the message with a query that

Algorithm 1 REQUEST_OBJECT

(**input**: agent A_i, search S constant K
output: boolean end)

high_probability_agents = {}
unknown_probability_agents = {}
while $end \in S = false$ **do**
  max_agents_to_request ← return_max_agents_to_request($maxt \in S, maxc \in S$)
  while max_agents_to_request > 0 **do**
    high_probability_agents ← return_high_probability_agents(K, $PR \in S, EOWT \in A_i, maxt \in S, maxc \in S$)
    if $|high\_probability\_agents| = 0$ **then**
      break
    for all $high\_probability\_agents_j \in high\_probability\_agents$ **do**
      update_estimation_willing_load($eowt_{j,i}, -C$)
      ask $obj \in S$ from $high\_probability\_agents_j$
    wait CONSTANT
    $max\_agents\_to\_request \leftarrow max\_agents\_to\_request - |high\_probability\_agents|$
  unknown_probability_agents ← return_unknown_probability_agents($PR \in S, EOWT \in A_i, maxt \in S, maxc \in S$)
  for all $unknown\_probability\_agents_j \in unknown\_probability\_agents$ **do**
    update_estimation_willing_load($eowt_{j,i}, -c$)
    ask $unknown\_probability\_agents_j$ if has $obj \in S$
return $end \in S$

Algorithm 2 PROCESS_ANSWER

(**input**: agent A_i message)

search S ← get_search($message$)
responder ← get_responder($message$)
$pr_{responder} \in PR \leftarrow get\_response(message)$
$eowt_{responder,i} \leftarrow get\_estimation\_of\_willingness\_of\_responder(message)$
if $get\_obj(message) = true$ **then**
  $end \in S \leftarrow true$
  update_willing($wt_{i,responder}, +C$)
else
  update_willing($wt_{i,responder}, +c$)

is sent (line 1), and determines the agent that sent the message (line 2). It then updates the probability that the responder has the requested object and its estimate of its willingness to send it (line 3–4). In line 5 it checks whether the object is received. If so it terminates the search by setting end to $true$. Finally, the agent updates its own willingness with respect to the responder. If the responder sent the object it increases its willingness by C, and if the response is a message then it increases the willingness by c. This process is executed in a thread, where the responder gets from the agents the information about the probability that they have the picture (line 3), so it knows which agent has the picture.

8.2 The Response Process

Once a request message for an object is received by an agent it should decide how to respond to it. This decision is in the core of this process. The agent will take into consideration two parameters: The current load of the agent ($ld \in A_i$), and its willingness to assist the requester agent ($WT \in A_i$). These parameters will determine whether to respond to the request at all, and what kind of response to send.

Algorithm 3, ANALYZE_REQUEST describes the response process. It is triggered with a request message received by the agent. In line 1 the agent extracts the requester agent from the request. It computes its regular load/willingness value in respect to the requester (load_willing_for_requester; line 2). Given an agent i, this computation depends on i's current load ($ld \in A_i$) and its willingness to assist the requester ($wt_{i,requester}$). In the experiments, we used $wt_{i,requester} - ld$) as the computation for load_willing_for_requester.

If the load_willing_for_requester value is greater than 0 , the agent checks that the request is for an object, that it indeed has the object, and that its current value of the load_willing_for_requester is greater than the estimated cost to send the object (line 5)[1]. If this condition is satisfied then it sends the objects (line 6), otherwise it sends the requester a message that it has the object and also the load_willing_for_requester value in order to let the requester update its estimate of the willingness of the agent to send the object (line 8).

However, if the value is smaller than or equal to 0 then the agent sends neither the object nor any message that it has the object (line 13–14), i.e.., it does nothing. Indeed, we would recommend reversing the test condition to in optimized versions of the code. We are explicitly noting the choice to do nothing in the pseudo-code.

Finally, A_i updates its willingness in respect to the requester, if it sent an object it reduces its willingness by C, and if it sent a message then it reduces the

[1] In the experiments, the cost was computed as the size of the object.

willingness by c. It also updates the estimation of the willingness of the requester to send it objects in the future as follow: If it sent the requester an object then it increases its estimation about the willingness of the requester by C, and if it sent a message then it increases it by c.

Algorithm 3 ANALYZE_REQUEST
(**input**: agent A_i, request)

1: requester ← get_requester(request)
2: load_willing_for_requester ← return_load_willing_for_requester($wt_{i,requester}, ld \in A_i$)
3: **if** load_willing_for_requester > 0 **then**
4: object ← get_object(request)
5: **if** $request = request\_for\_object \wedge have(object) \wedge load\_for\_requester \geq return\_cost\_of\_sending(object)$ **then**
6: send(load_willing_for_requester,$object$)
7: update_willing($wt_{i,requester}, -C$)
8: update_estimation_willing_load($eowt_{requester,i}, +C$)
9: **else**
10: send(load_willing_for_requester,have(object))
11: update_willing($wt_{i,requester}, -c$)
12: update_estimation_willing_load($eowt_{requester,i}, +c$)
13: **else**
14: do nothing

The willingness thresholds are determined by the user (i.e., by the application using the search processes). The update function for thresholds can also be determined by the user, i.e., the formula of how fast the willingness decreases/increases (as well as the load) could be changed. For example, linearly or polynomially in the number of requests, changing with time, etc. However this formula should be known by all members of the group. The estimation thresholds of the others are set according to initial value. The agent updates the estimation thresholds in several ways:

1. Based on the data in the messages that the agent receives, when querying for their state.

2. Based on expected decrease in load, as time passes.

3. Based on requests sent to other agents. For each request, we expect the load to go up, and the willingness to go down.

8.3 A Complete Run through the Search Process

In Figure 8.1 we show the parallel time-lines illustrating a complete run of the processes, for a single search. The search begins with specific query from the user. The agent starts a new search process, by deciding on candidates who are likely to respond; this is done based on their current willingness to respond (referred to as EOWT, associated with each agent), which is inversely proportional to their estimated load. Before it sends the message, it decreases the EOWT of the other agents, as they are now estimated to be loaded with the query (and thus their willingness to respond again will be lower). The other agent analyzes the request according to its true willingness to help the agent (WT) and then it checks that it has the object, updates the willingness and sends a response that it has the object and its new state. The agent analyzes the response, updates its estimate of the other agent and that it has the object. After that, the search process recognizes that the other agent has the object and that it is able to send the object, it updates the estimation (reduces it, because it is going to ask it for the object) and asks the other agent to send him the object. The other agent analyzes the request and according to the willingness, it decides to reduce the willingness and to send the object and the state. The agent analyzes the response object and state, displays it to the user, updates the estimate and terminates the search.

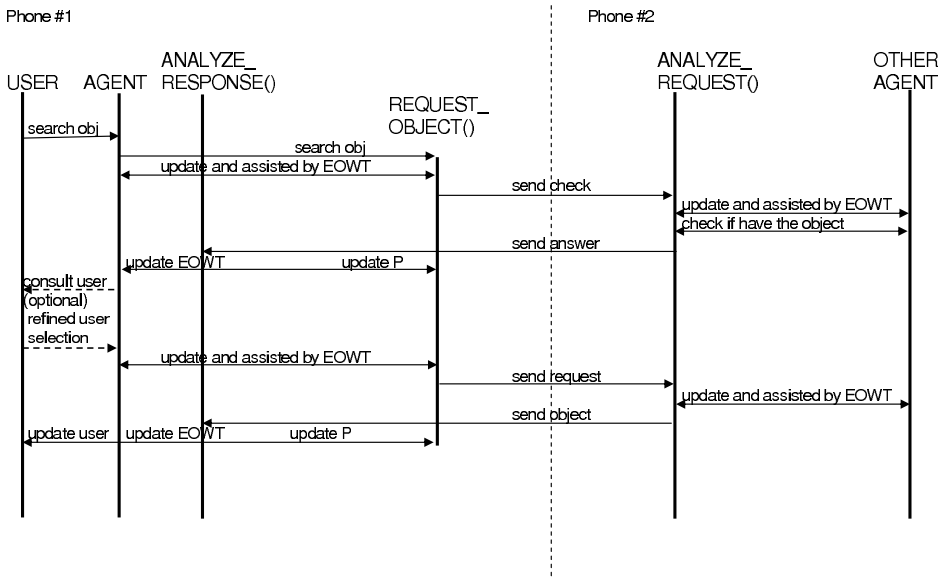

Figure 8.1: Search process time-lines.

8.4 Two variant algorithms

The following parameters can be set by the user, or using default values.

$maxt$ determines the time limit place on the search. Smaller values will force speedier search times.

$maxc$ determines the maximum cost incurred by the search process through messages. Smaller values will force fewer messages (and thus lower costs).

By tweaking these two thresholds, one can create variants of the algorithm that would be faster and fairer (lower $maxt$ values), but try to send a greater number of messages; or variants that would be cheaper in terms of messages sent (lower $maxc$) values, but may therefore take longer to complete the search.

We distinguish these two parameters to allow fine control over the behavior of the agent. The two parameters determine which agents will get into the $less\_probability\_agents$ and $high\_probability\_agents$ lists. Setting both of the thresholds to low values will yield the qualitative behavior of an agent easily giving up on searching, while refusing to send many messages out. Setting up both of the thresholds to high values will yield behavior similar (in the limit) to brute-force searching.

We will explore two qualitative variants of the algorithm:

Fast Walk (*FAST*). This variant has low $maxt$ and high $maxc$. It thus prefers cutting down search time, even at the cost of additional messages.

Cheap Walk (*CHEAP*.) This variant has higher $maxt$, and lower $maxc$. It thus prefers sending fewer messages.

Chapter 9

Experimental Design and Analysis

9.1 Experiment Design

To evaluate the capabilities of our algorithm we used a simulation of a cellular network. The basis for the simulation is provided by STRI. We have modified it to account for the passage of time during transmissions. In addition, we simulate users in closed groups (e.g., family or group of friends). Each user can have different willingness thresholds, associated with different peers. Each user has a database of pictures, and its estimates of others' willingness.

The agent represents the user, and its goal is therefore to save search time and communication costs. The agent has access to the picture database of the user. We assume that the agent knows which pictures are approved for sharing by the user. The agent knows all the other agents in the user's closed group. We additionally allow the agent access to the device itself, so it can know the load of the communication and the level of the battery.

We contrast three algorithms: The FAST and CHEAP algorithms described above, and the classic random walk (RAND) algorithm, which serves as a baseline. In this implementation of RAND, the search agent queries four randomly-chosen agents for the picture. It waits a fixed time duration for responses before continuing to the next four random agents. The first agent that sends a positive response is requested to transmit the picture. The search agent does not wait to get the picture: It continues the query process until it receives an announcement that the picture is being sent.

For the evaluation of the FAST and CHEAP algorithms, we set the $initial$ value to 10 (initialize the willingness thresholds and the estimation thresholds to 10). Updating the willingness as a result of sending a picture was set to +/- 1 (the constant C):

1. Minus one (-1) for the sender.

2. Plus one (+1) for the receiver.

This means that the receiver will have more willingness to help the sender next time and the sender will have less willingness to help the receiver next time. However, a request for information affects the willingness of the receiver, only at a tenth of the effect of sending a picture, i.e., by +/- 0.1 (the constant c). In addition, each request for information increases the estimation about the load of the receiver by one and each request for a picture increases by the average size of the pictures. Assume $eowt_{j,i}$ is A_i's estimation of agent A_j willingness threshold (composed of its willingness and load), with respect to agent A_i. Assume that A_i sends a request for information to A_j, then $eowt_{j,i} \leftarrow eowt_{j,i} - 1$, and if A_i sends a request for a certain picture to A_j, then $eowt_{j,i} \leftarrow eowt_{j,i} - C_{pic}$, where C_{pic} is an application-dependent constant. In the experiments, C_{pic} was set to the picture size used. In practice, the value of C_{pic} should be determined experimentally, for instance by setting it to the expected average picture size or to other values.

Other Independent Variables. We simulated requests for pictures of various sizes, in various group sizes. The duplication of pictures within the group was controlled as well.

Thus the independent variables included:

1. Number of agents in a group: 30, 50, 70, 100.

2. Size of pictures: 2, 4, 8 MB.

3. The probability that any one agent (independent of others) will have a certain picture from one general pool: 0.05, 0.1, 0.2, 0.4, where all the pictures have the same probability. Note that due to the independence assumption, these rates are uniform for the population of users. So setting up a rate of 0.4 probability of having a picture, implies that this picture will be present in 40% of agents in the group.

We did not test all combinations, but instead always fixed one or two of the values, while varying the others. The default fixed values were 30 and 70 for the group size, 4MB for the message size, and 0.2 for the probability of having a picture. See below for details. In order to simulate load requests in the network, every user searches for four pictures in average per experiment. Each search is conducted for one picture, and then in sequence another search can be performed. Every configuration is repeated 30 times. This is done to control for confounding effects due to simulation randomness.

Dependent Variables. We recorded the values for the following variables:

1. The time that it takes to retrieve a requested picture (in seconds).
2. The number of sent messages to retrieve a requested picture.
3. The agents that sent pictures.

By analyzing this data we could evaluate our algorithm in terms of efficiency and fairness. The efficiency of the algorithm is measured by the time and the load to acquire a picture. The fairness is measured by the distribution of sent pictures among the group's members. The more uniform the distribution is the more fair the algorithm is.

We analyze experimental data to evaluate efficiency and fairness. The efficiency of the algorithm is measured by the runtime and the load to achieve a picture. The fairness is measured by the distribution of sent pictures among the group's members. The more uniform the distribution is, the fairer the algorithm is.

The time to acquire a picture is measured from the moment the agent asks for it, until the moment that it receives it in full (*retrieve time*). We compute the average case, the average worst case and the average standard deviation, which is here taken to be the average of the standard deviation of all the experiment in a configuration. As agents send pictures one after the other, the standard deviation is high relative to the average. Average maximum retrieve time is computed as the average over the maximum time measured for each experiment in a configuration.

The load of acquiring a picture is measured as the number of messages sent until finding an agent that has the picture and will transmit it. The load is marked *number of checking* (number of queries) in the graphs below. We similarly compute the average load, the average maximum load, and the average load standard deviation.

Fairness is measured by average standard deviation of sending pictures (*standard deviation of sending Pictures*).It represents the distribution of sent pictures among the group's members. Low standard deviation of sending Pictures means high fairness, as the transmission load across agents is closer to the average.

In general, the results show that the FAST and CHEAP algorithms are fairer than the random walk (RAND) algorithm and often faster. The FAST algorithm is always faster, but is also more expensive in terms of query messages. The CHEAP algorithm is less fair than FAST, but still has better retrieval time than RAND in most cases, and often uses less messages than RAND. The exceptions take place in environments with few agents and with rare pictures. Here, RAND uses less messages, but it will always take more time to retrieve them, in comparison to CHEAP or FAST.

9.2 Results and Analysis

9.2.1 Influence of Group Size

We first discuss experiments addressing the influence of group size on the performance of the algorithms. In all configurations here, the probability an agent will have a certain picture from one general pool is 0.2, and the picture size was set to 4MB.

Figure 9.1 shows that in all group sizes, the FAST algorithm used more messages for retrieving pictures. The difference between the FAST algorithm and the other algorithms is significant (paired t-test, $p < 0.0001$) and pronounced (more then 53%). The FAST algorithm used many messages to evaluate the state of the others. Up until 70 agents the amounts of messages increase because there are more agents to evaluate, with the goal of finding an available agent, where the meaning of available here is an agent that has the picture with high willingness to send it. Above 70 agents this mission is easier for two reasons: (1) there are more available agents; and (2) more agents per every query can help a querying agent evaluate their state.

For larger groups (70 and above), there is no significant difference between the algorithms. For small groups (less then 70 agents) the CHEAP algorithm used more messages than RAND. But the difference between the CHEAP and the RAND algorithms is significant (paired t-test, $p < 0.003$) but not pronounced (less then 6%).

Figure 9.3 and 9.2 also demonstrate that the FAST algorithm significantly used (paired t-test, $p < 0.001$) more messages to retrieve a requested picture. Figure 9.2 presents the average worst case (maximum time) for retrieving one picture over 30 trials. In the worst case there is no significant different between the CHEAP and the RAND algorithms. The difference between 70 agents and 100 agents in the FAST algorithm is not significant, which means that as the number of agents in the group increases, the worst case also increases. In figure 9.3 there is significant difference (paired t-test, $p < 0.02$) between CHEAP and the RAND algorithms only for a group of 30.

Figure 9.4 contrasts the fairness of the algorithms. As the standard deviation is lower, the distribution of the load of sending pictures is more uniform, and therefore fairer. There is significant difference (paired t-test, $p < 0.0001$) between all the algorithms, with FAST being fairer than CHEAP, which is fairer than RAND. Indeed, the fairness of the RAND algorithm does not change in a significant way up until 70 agents. The reduction between 70 to 100 agents in the RAND algorithm is significant (paired t-test, $p < 0.0001$). It shows that in larger groups, randomizing works better than in small groups. There is also significant (paired t-test, $p < 0.004$) reduction in standard deviation (i.e., an increase in fairness)

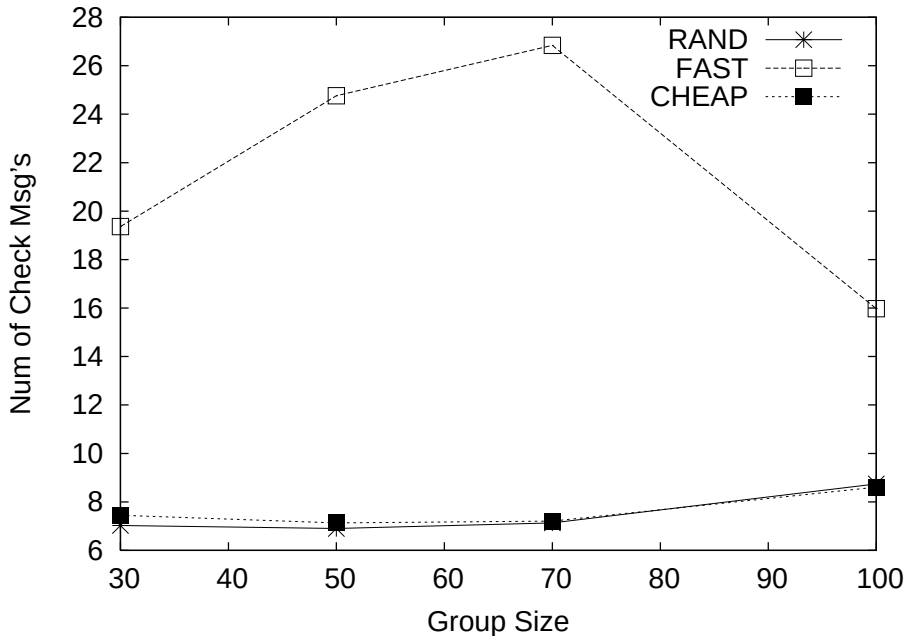

Figure 9.1: A comparison between the average number of check messages as a function of group size

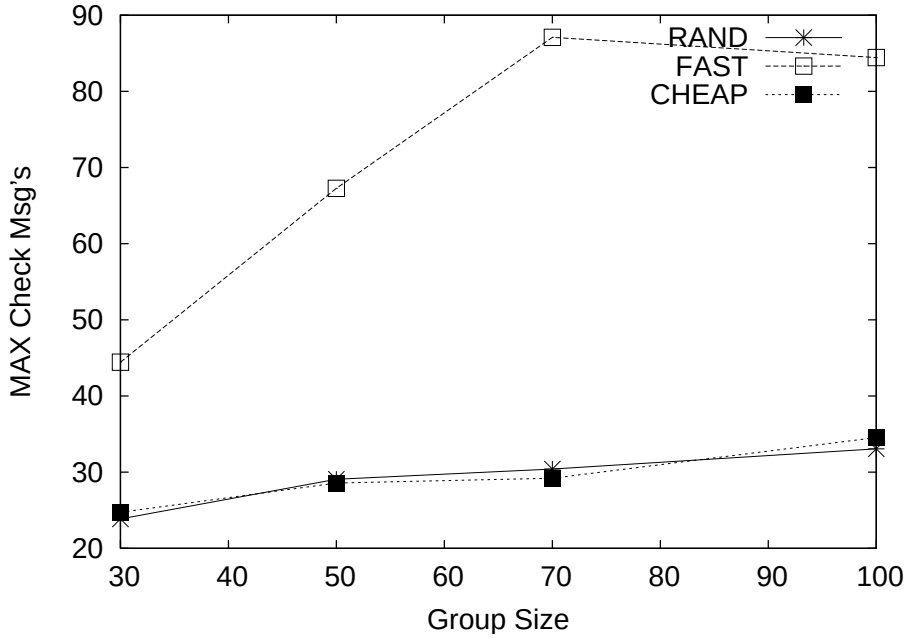

Figure 9.2: A comparison between the average maximum number of check messages as a function of group size

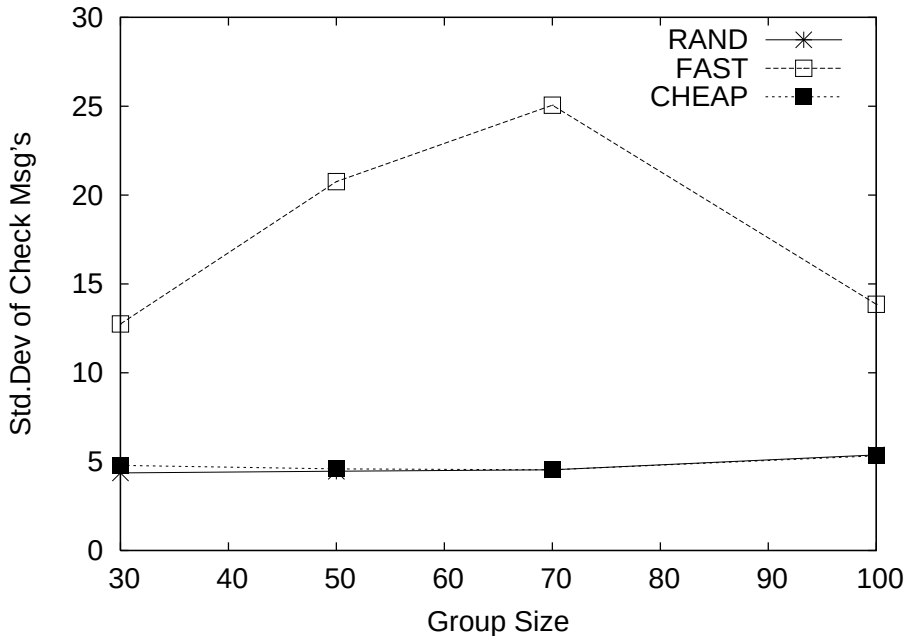

Figure 9.3: A comparison between the average standard deviation of querying as a function of group size

between 30 to 50 agents in the FAST algorithm. For these group sizes, FAST can easily find agents that (1) have the picture and that (2) are not utilized unfairly. In contrast, the CHEAP algorithm increases its standard deviation, because it places heavier emphasis on minimizing the amount of messages, than on fairness.

A comparison of Figure 9.4 to Figure 9.5 reveals that algorithms that are fairer retrieve pictures faster. Figure 9.5 validates the hypothesis that as the number of agents in group increase it takes more time in average to retrieve pictures, because there are more cases where one agent sent many pictures and therefore delivered them slower. This explains the correlation between fairness and retrieval time. Fairness means that each agent sent lees pictures and therefore the retrieval is faster (less pictures that the sender sent in parallel). In Figure 9.7 and 9.6 the significant difference between the algorithms shows that in the worst case and in all the range of deviation, the relationship between algorithms remains invariant.

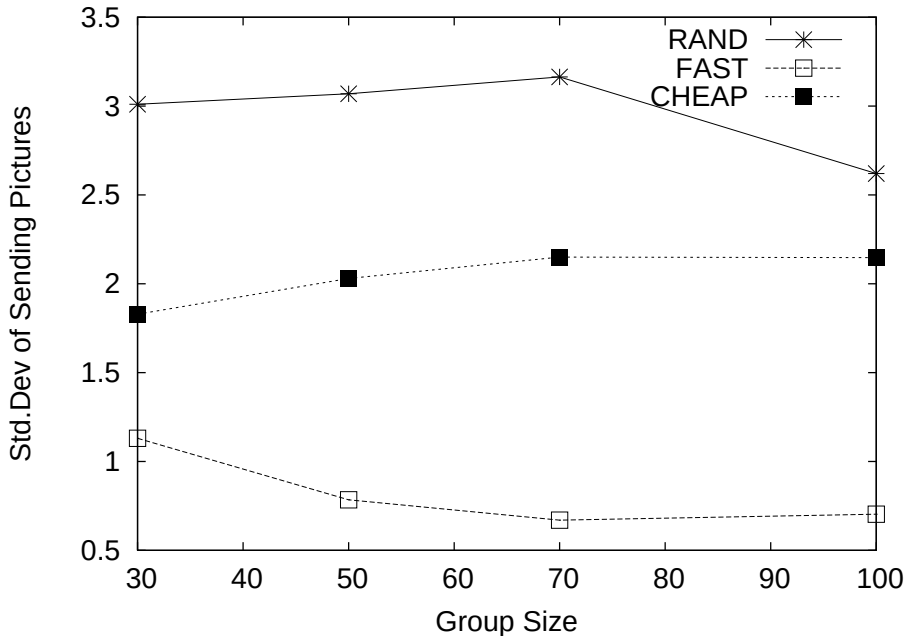

Figure 9.4: A comparison between the average standard deviation of sending pictures as a function of group size. Lower standard deviation implies greater fairness.

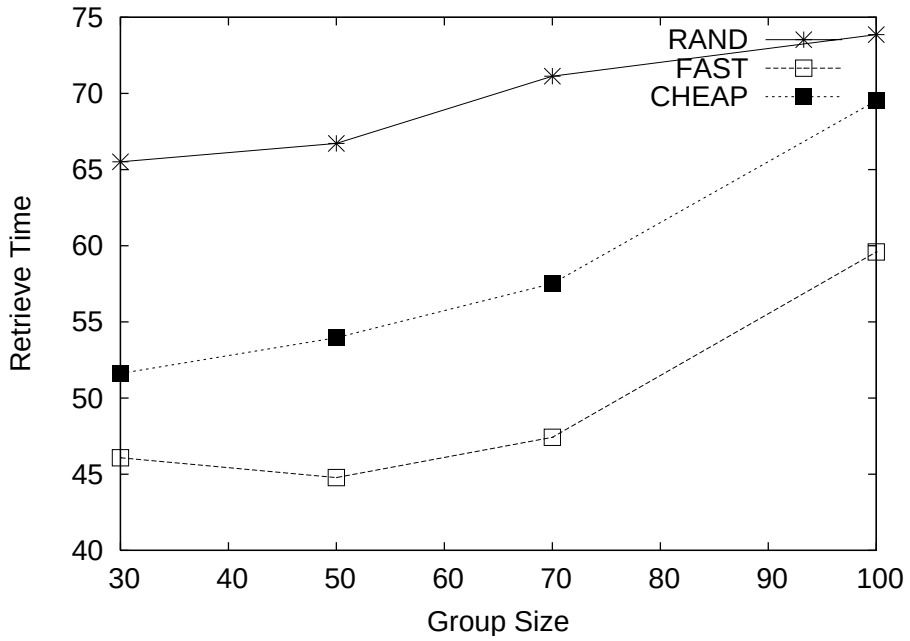

Figure 9.5: A comparison between the average retrieval time as a function of group size

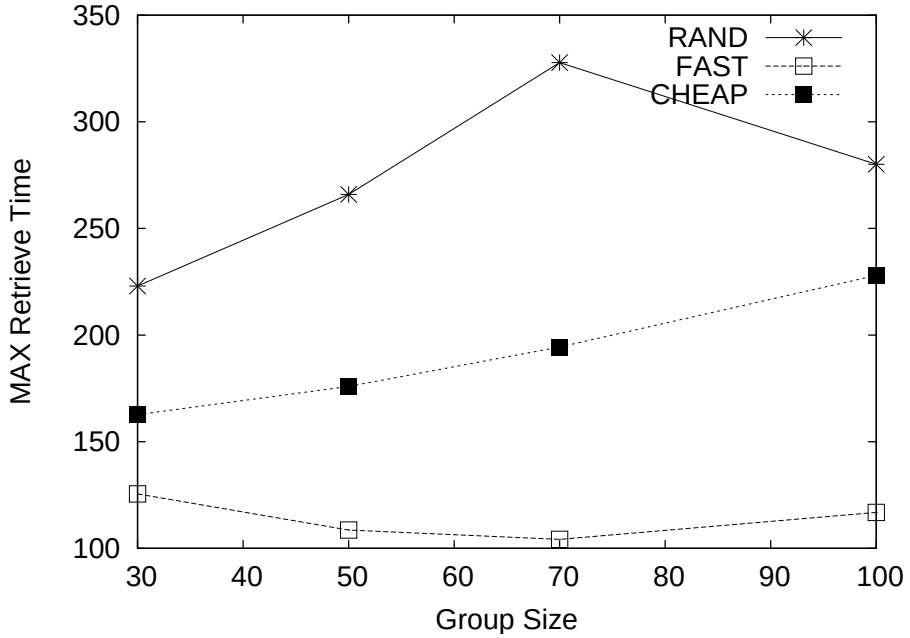

Figure 9.6: A comparison between the average maximum retrieval time as a function of group size

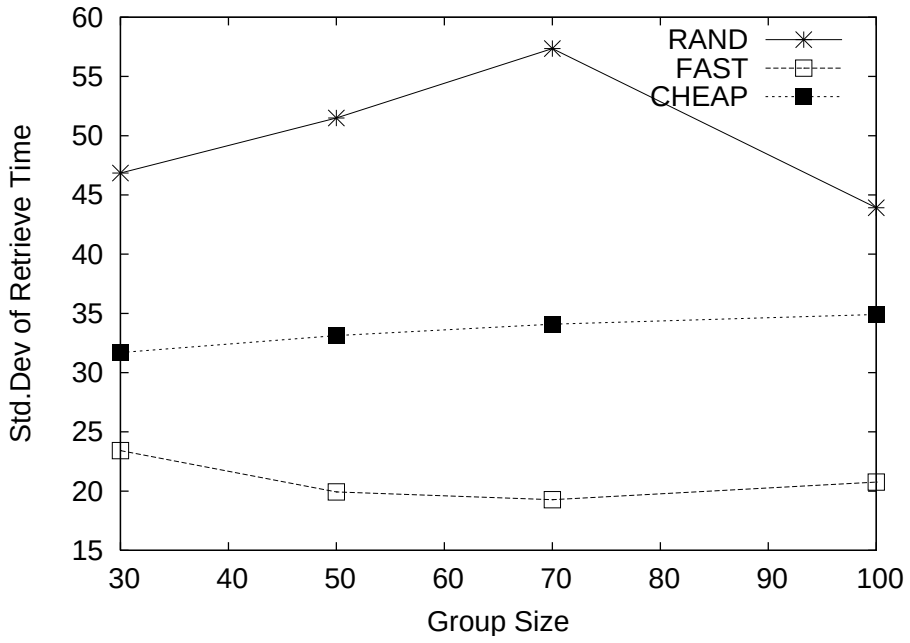

Figure 9.7: A comparison between the average standard deviation of retrieval time as a function of group size

9.2.2 Influence of Pictures Size

We now turn to examine the influence of the picture size. In all the configuration reported in this section, the probability of having a picture was 0.2. We examine two settings for the group size: Large groups (70 agents), and small groups (30 agents). Both are evaluated with picture sizes of 2, 4, and 8MB.

The size of the picture is not supposed to influence the number of messages that it takes to retrieve it. However, in our algorithm we used messages to also evaluate the state of other agents. Therefore we might use more messages, as larger pictures cause the FAST algorithm to more carefully consider whom to download from, as explained below.

Large Groups

Figure 9.8 shows that the the RAND and CHEAP algorithms essentially maintain constant (and equal) numbers of queries across picture sizes. The reason for the similarity is that both stop searching once they find an agent that has the picture. The CHEAP algorithm may sometimes continue querying looking for an agent that is also available, but in larger groups this is easier to find, and thus there are no difference in performance between the two algorithms.

In contrast, the FAST algorithm does not stop the search until it finds an *available* agent that has the picture and that can transmit it quickly (i.e., is not already engaged in transmission). As the size of pictures increases, the agents that send them are busier, and therefore it takes more time to find an available agent. Figures 9.9 and 9.10 describe the average maximum number of queries sent, and the average standard deviation, of the number of messages sent to retrieve a requested picture.

The size of the picture influences the fairness, and therefore the the retrieval time. As the size of the pictures increases, the difference between the number of the pictures that each agent sends is more significant. Figure 9.11 shows a slight, but significant (paired t-test, $p < 0.04$) increase in the average standard deviation of the picture transmission in the FAST and CHEAP algorithms. The RAND algorithm is not influenced in a significant way, from different sizes of pictures.

The significant influence of the picture sizes on the retrieval time is obvious, as larger pictures take longer to transmit. As the size increases it takes more time to transmit (retrieve) the pictures. However, the increased fairness makes our algorithms (FAST and CHEAP) faster than the RAND algorithm. In figures 9.12 , 9.13 and 9.14 it seems that the gap between the algorithms increase as the size of the pictures increase. The FAST algorithm is faster than the CHEAP algorithm because the requesting agent uses more messages in order to find the agents that are most willingness to transmit pictures, and so it receives the object faster.

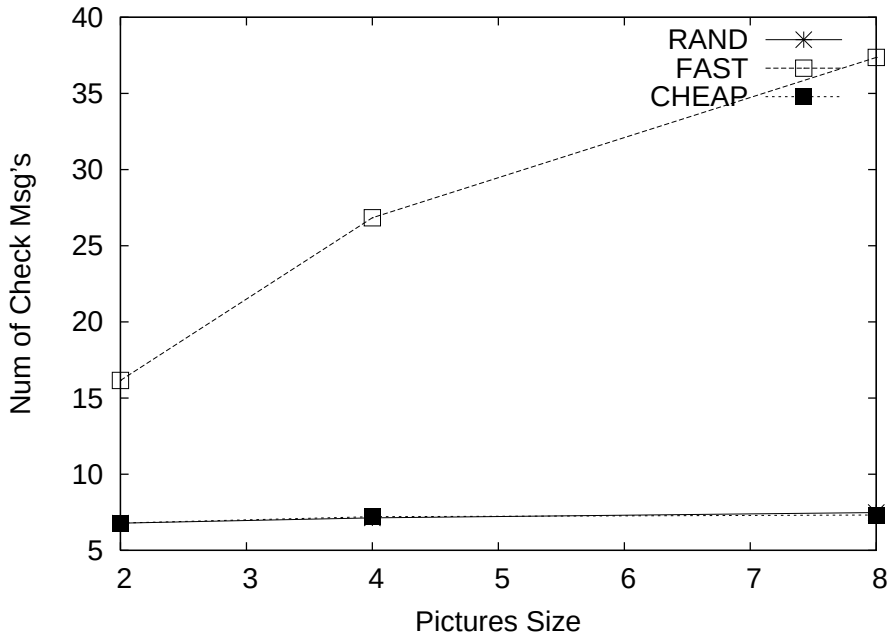

Figure 9.8: A comparison between the average number of check messages as a function of pictures size in big group

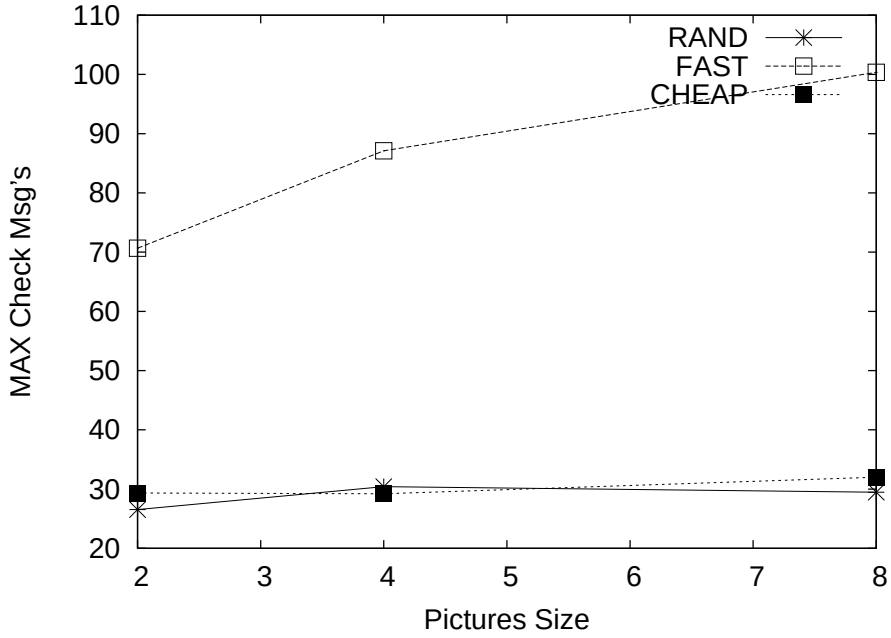

Figure 9.9: A comparison between the average maximum number of check messages as a function of pictures size in big group

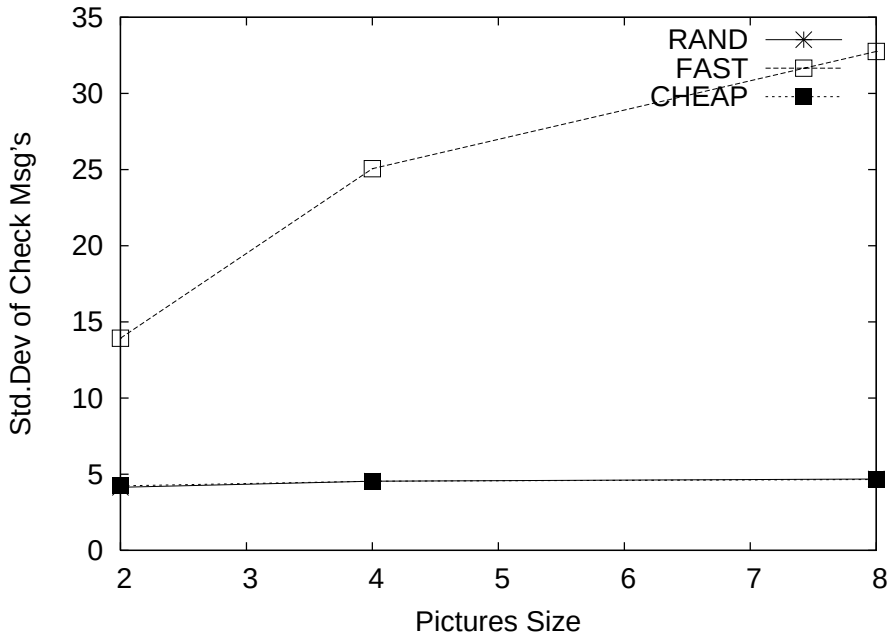

Figure 9.10: A comparison between the average standard deviation of querying as a function of pictures size in big group

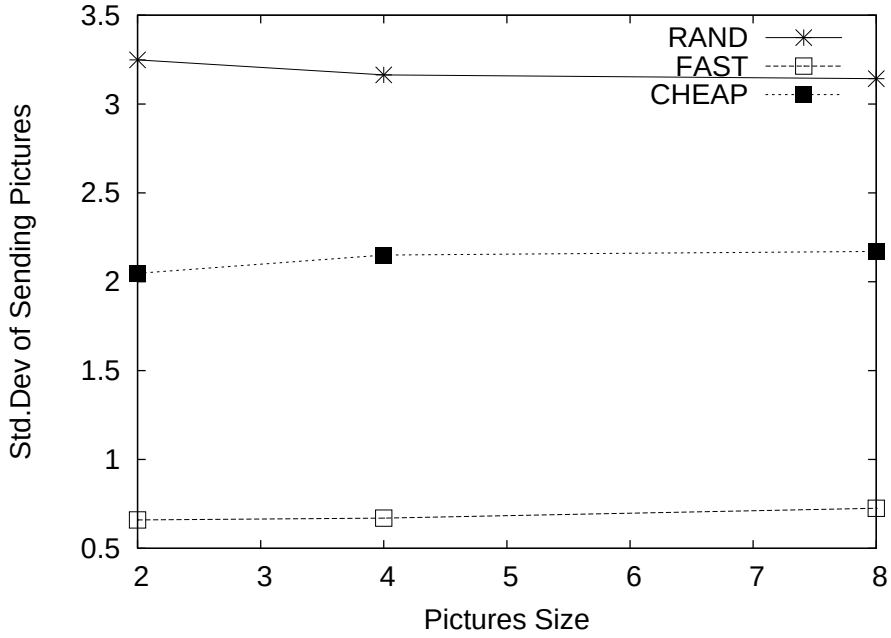

Figure 9.11: A comparison between the average standard deviation of sending pictures as a function of pictures size in big group

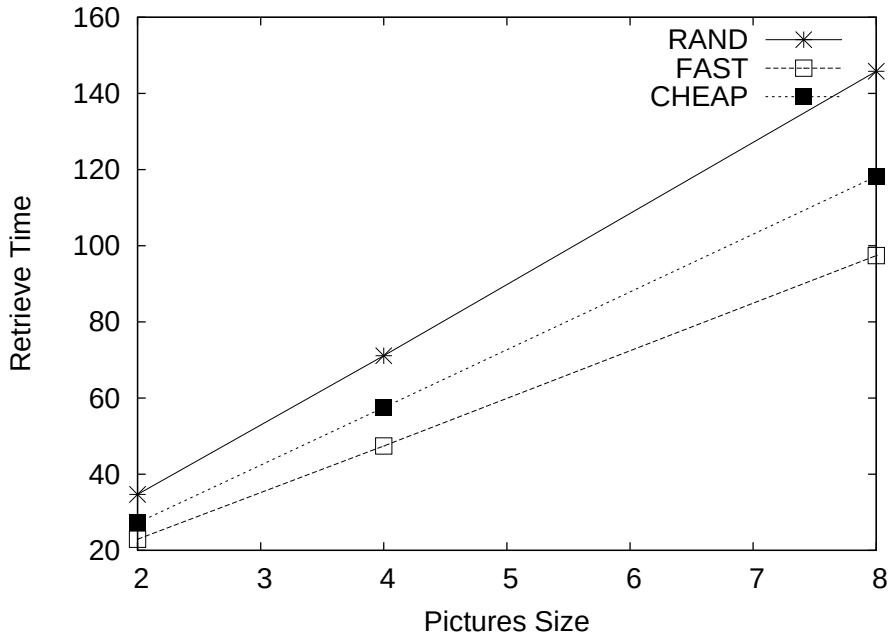

Figure 9.12: A comparison between the average retrieval time as a function of pictures size in big group

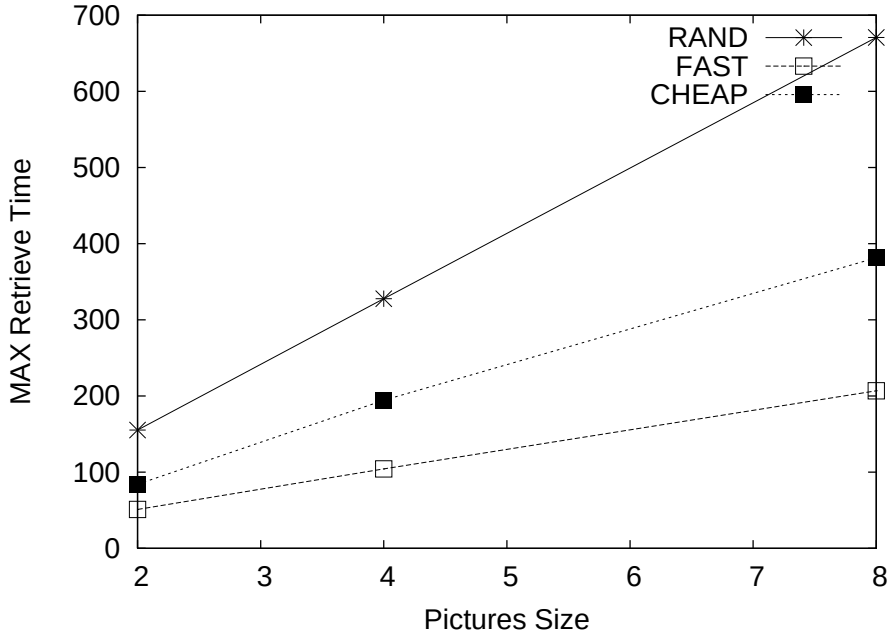

Figure 9.13: A comparison between the average maximum retrieval time as a function of pictures size in big group

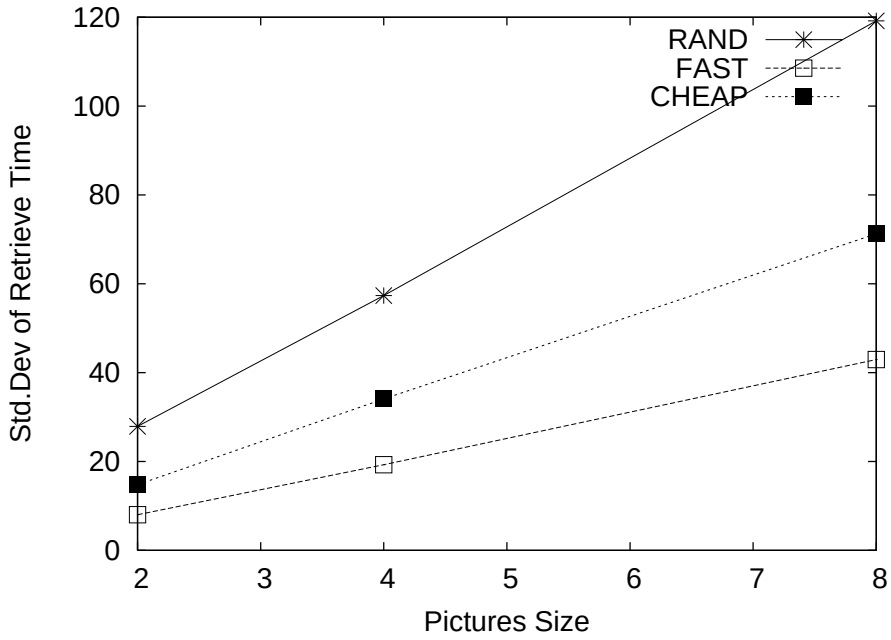

Figure 9.14: A comparison between the average standard deviation of retrieval time as a function of pictures size in big group

Small Groups.

The results in small groups are similar to those with large groups, but with a few notable differences, due to the size of the group.

First, while the number of queries remains low (and almost constant) for the CHEAP and RAND algorithms, there is still a difference between CHEAP and RAND, unlike in the case of bigger groups. The cause for this is that it is more difficult to find an available agent in a smaller group (even by the more relaxed criteria used by CHEAP). Thus CHEAP no longer stops querying after finding any agent that has the picture. Instead, it needs to continue querying a bit longer—and the difference with RAND becomes more pronounced (Figure 9.15). Above the size of 4MB, there is a significant (paired t-test, $p < 0.04$) difference between the RAND and the CHEAP algorithms. As the size increases, the difference also increases. Figures 9.16 and 9.17 describe the average maximum and the average standard deviation of queries sent until a picture was retrieved.

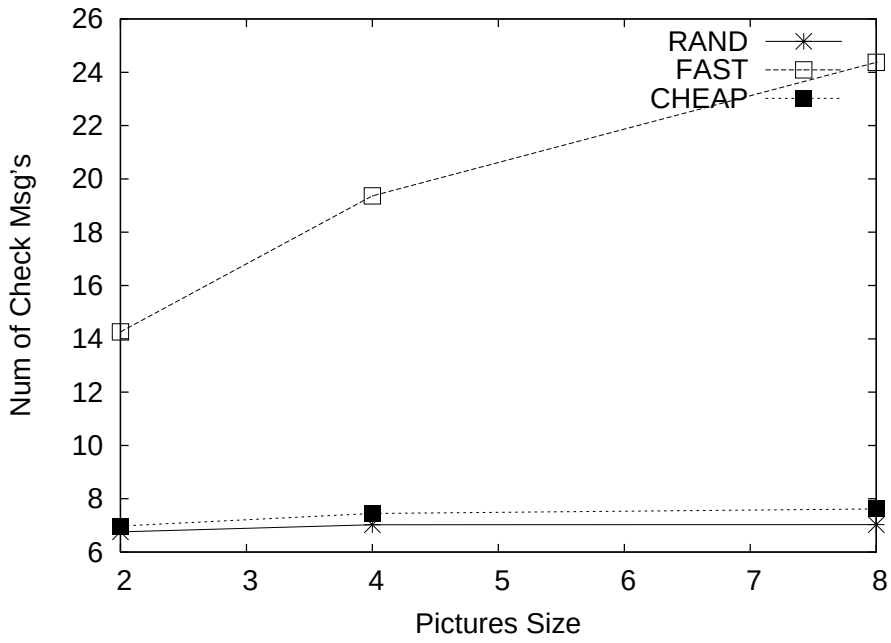

Figure 9.15: A comparison between the average number of check messages as a function of pictures size in small group

The results for fairness (Figure 9.18) are similar to the results for the large groups. However, the RAND algorithm shows a slight increase in standard deviation (reduced fairness), as compared to the larger groups. It does not reach the level of fairness achieved in the large groups because there are simply less agents to choose from.

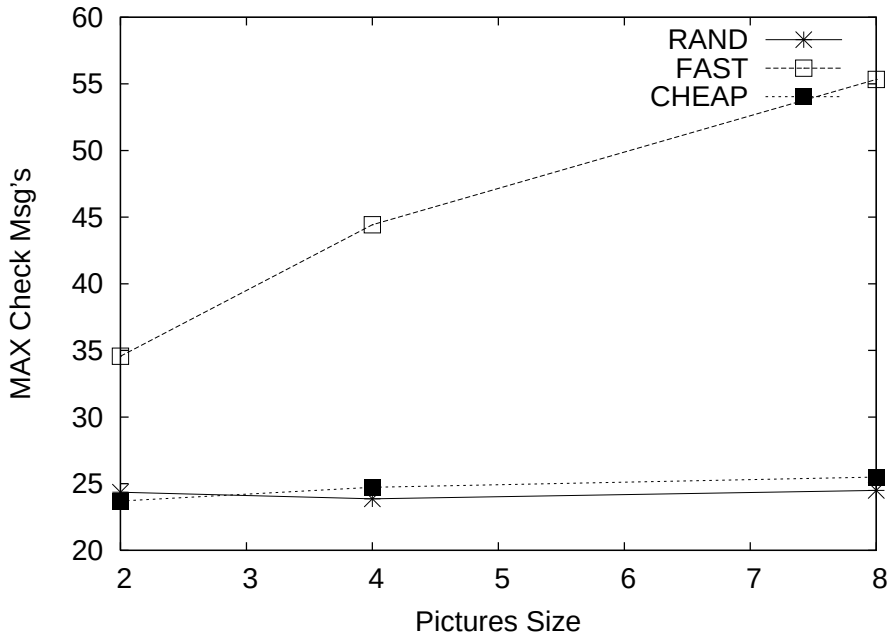

Figure 9.16: A comparison between the average maximum number of check messages as a function of pictures size in small group

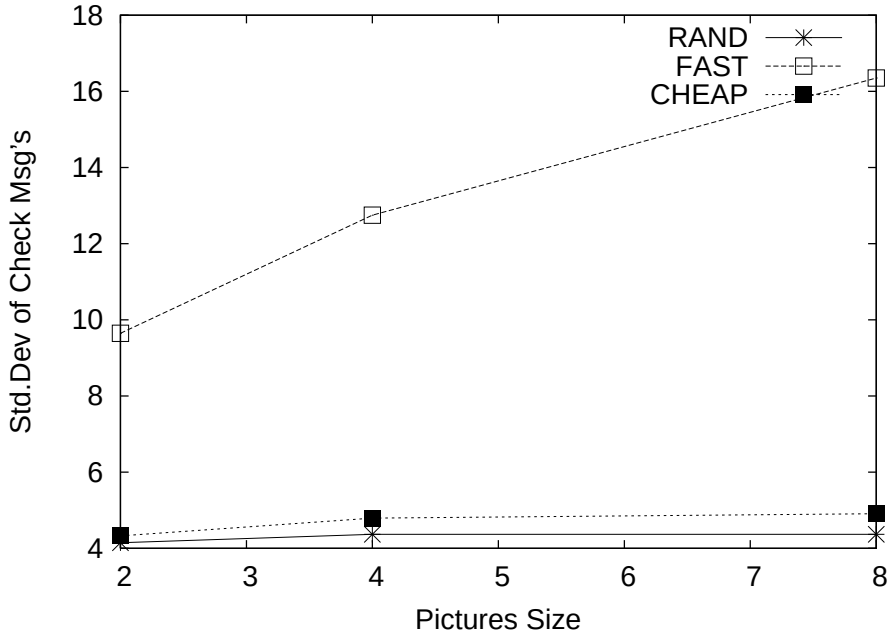

Figure 9.17: A comparison between the average standard deviation of querying as a function of pictures size in small group

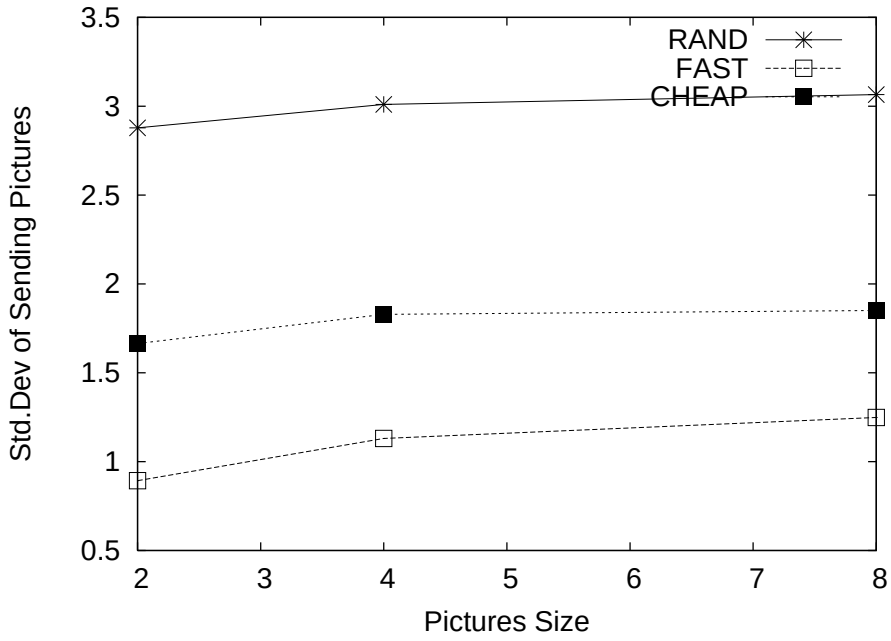

Figure 9.18: A comparison between the average standard deviation of sending pictures as a function of pictures size in small group

The results for the retrieval time are similar to the results in the large groups. See Figures 9.19, 9.20 and 9.21.

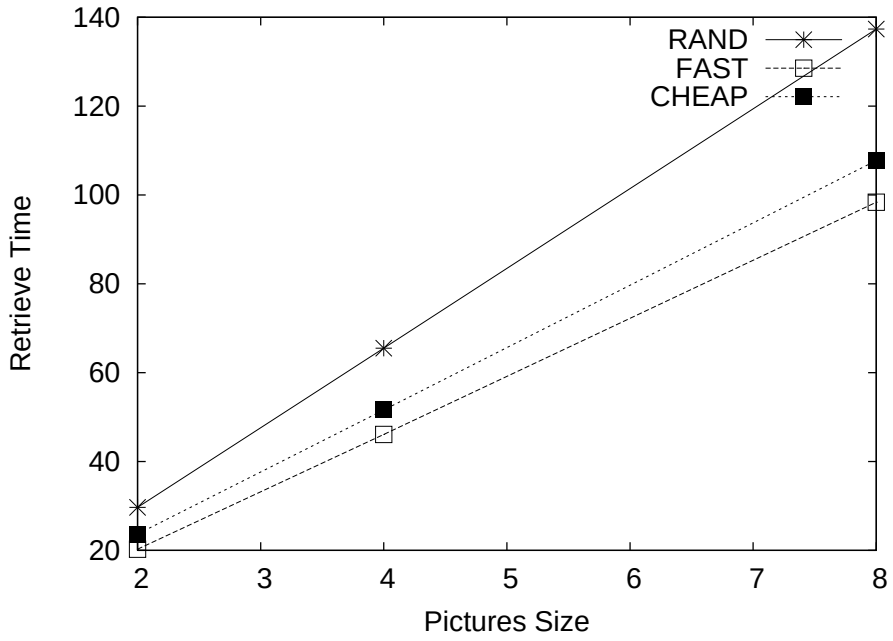

Figure 9.19: A comparison between the average retrieval time as a function of pictures size in small group

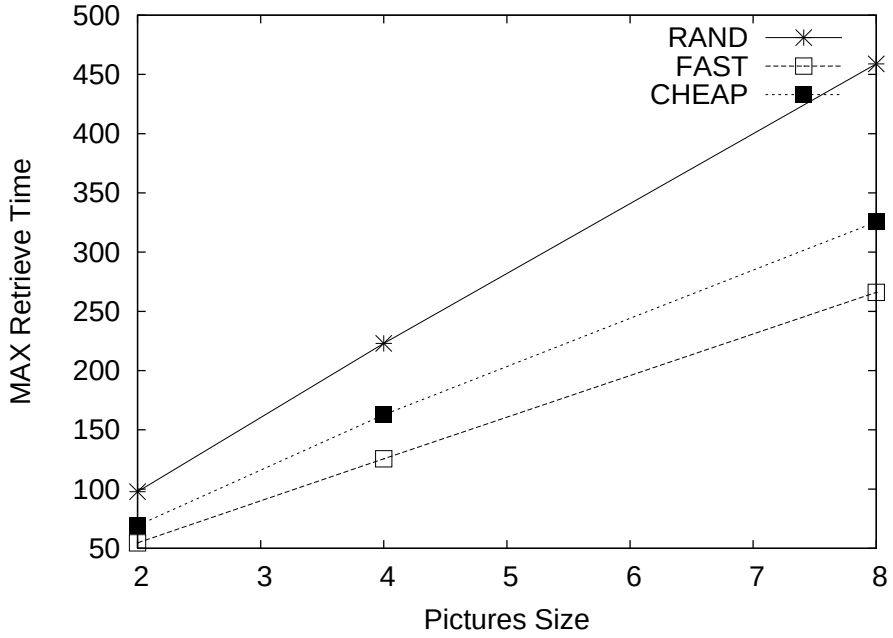

Figure 9.20: A comparison between the average maximum retrieval time as a function of pictures size in small group

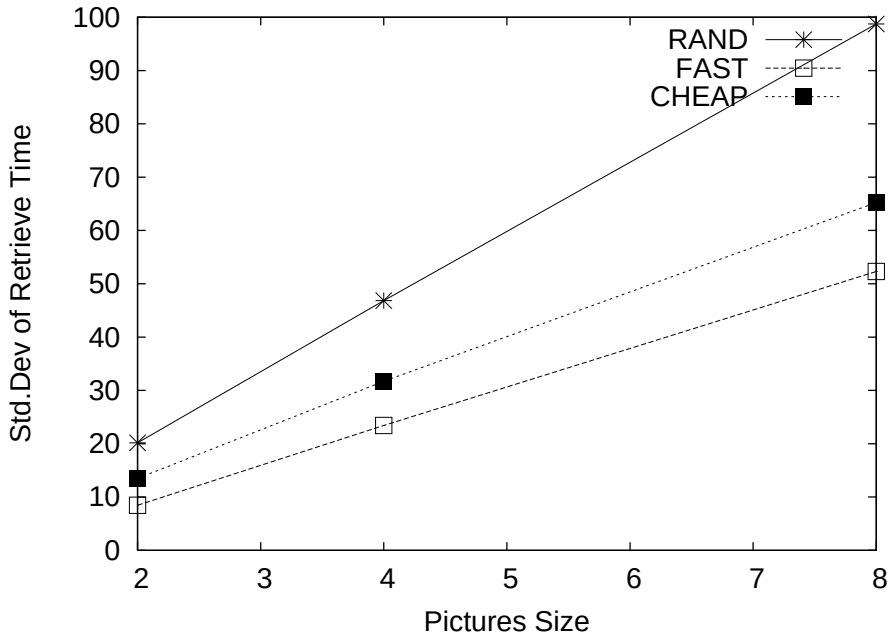

Figure 9.21: A comparison between the average standard deviation of retrieval time as a function of pictures size in small group

9.2.3 Influence of Picture Availability

We now turn to examine the influence of the picture availability. In all the configuration reported in this section, the size of the pictures is fixed at 4MB. We examine two settings for the group size: Large groups (70 agents), and small groups (30 agents). Both are evaluated with probability of 0.05 – 0.4 for having a picture.

Large Groups

Figure 9.22 shows how the percentage of agents that had a certain picture influence the average number of needed messages to retrieve the picture. As the percentage increases, more agents have each wanted picture and therefore it less messages are needed to find a source for the picture.

The FAST algorithm used significantly (paired t-test, $p < 0.001$) more messages to achieve fairness, but it is faster. Up until availability of 20%, the CHEAP algorithm used significantly (paired t-test, $p < 0.001$) more messages than the RAND algorithm. The reason for this inequality is that when availability is low, many agents are requesting a few pictures, that only a few agents have. The agents that owned these rare pictures are therefore busier, and their willingness to contribute is low. Since the CHEAP algorithm estimates this decrease (though not to the same degree as FAST), it only contacts these agents as a last resort. Figures 9.23 and 9.24 support this explanation.

Availability influences fairness significantly (Figure 9.25). There is significant difference (paired t-test, $p < 0.0001$) between all the algorithms, where FAST is fairer than CHEAP, and CHEAP is fairer than RAND. As availability increases, there are more agents that want a picture, therefore more sources to choose from, and thus it is possible to get fairer decisions. Above a probability of 20%, there is high probability to find agents that have the picture among the first four agents that are queried, and thus the fairness of the RAND algorithm remains constant. Note, however, that the performance of FAST and CHEAP is still superior.

Figures 9.26, 9.27 and 9.28 show similar relationship between the algorithms in the retrieval time. FAST is faster than CHEAP, which is faster than RAND.

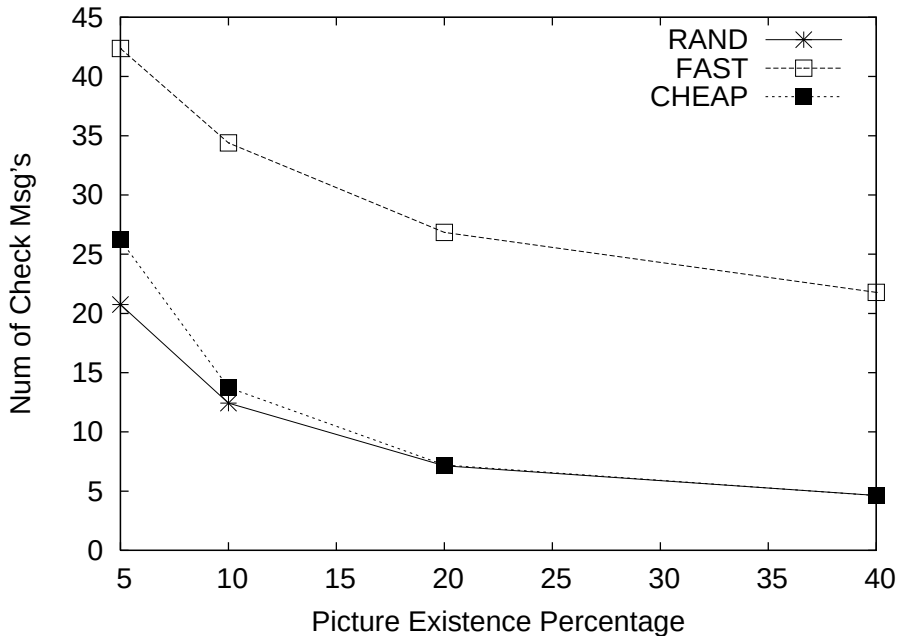

Figure 9.22: A comparison between the average number of check messages as a function of percentage in big group

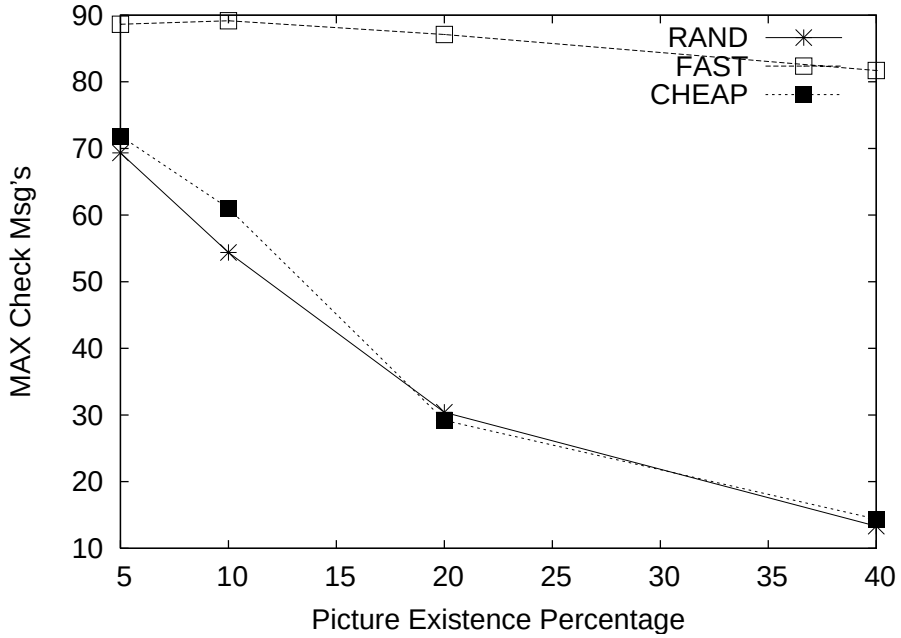

Figure 9.23: A comparison between the average maximum number of check messages as a function of percentage in big group

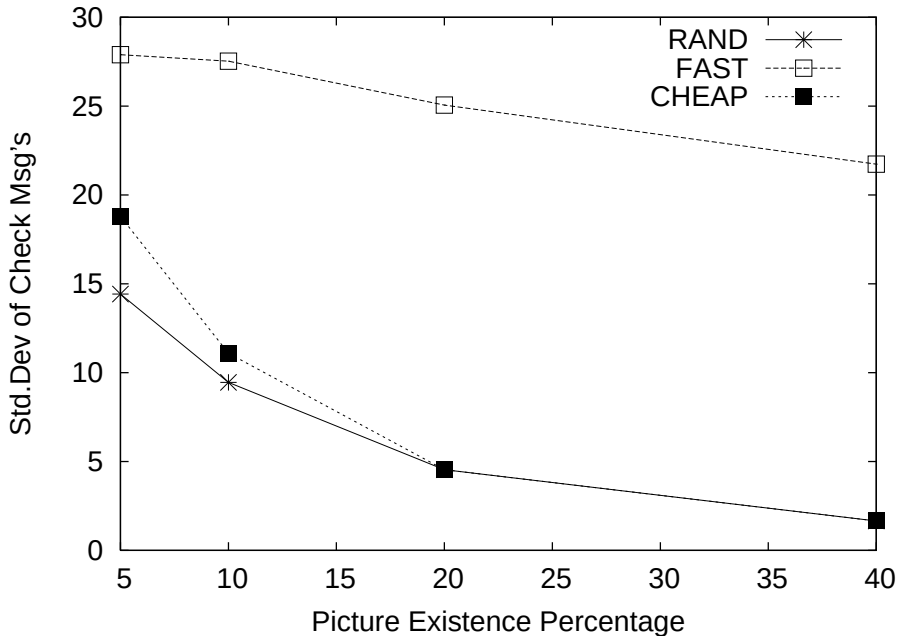

Figure 9.24: A comparison between the average standard deviation of querying as a function of percentage in big group

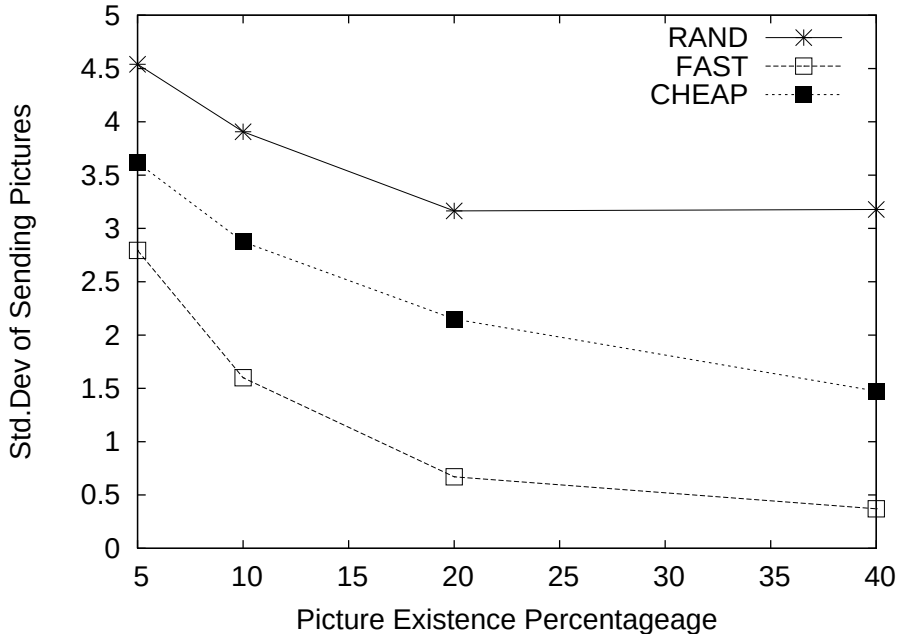

Figure 9.25: A comparison between the average standard deviation of sending pictures as a function of percentage in big group

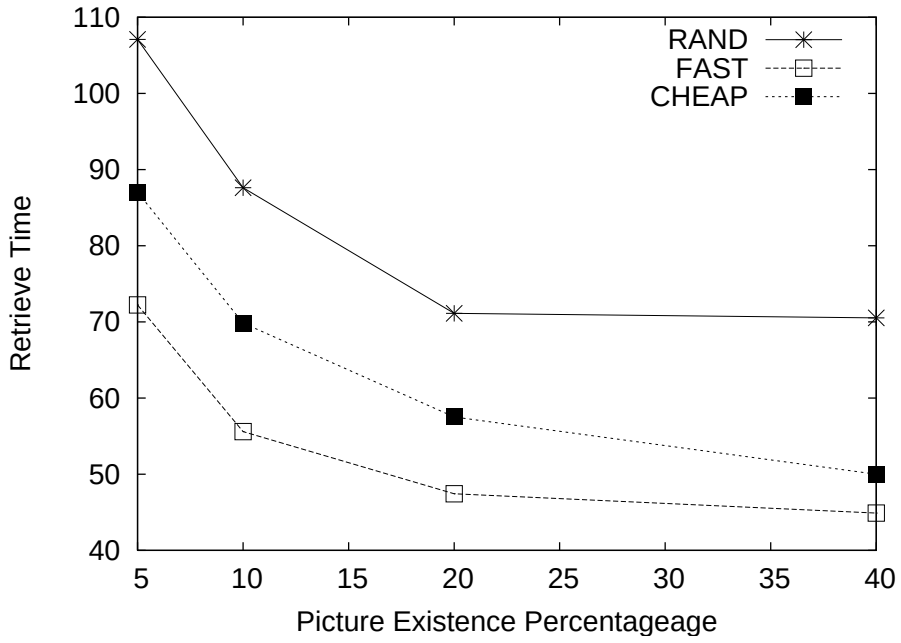

Figure 9.26: A comparison between the average retrieval time as a function of percentage in big group

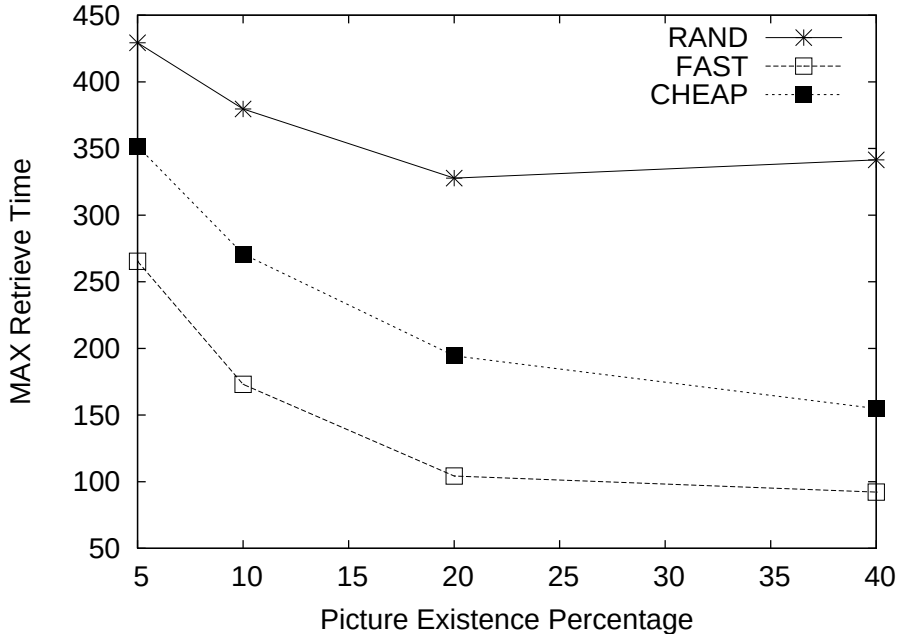

Figure 9.27: A comparison between the average maximum retrieval time as a function of percentage in big group

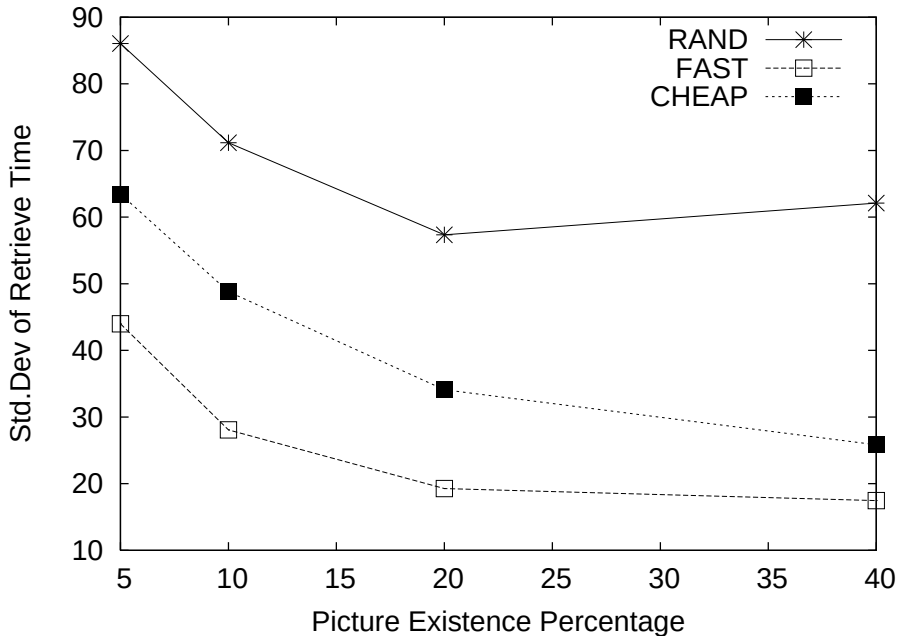

Figure 9.28: A comparison between the average standard deviation of retrieval time as a function of percentage in big group

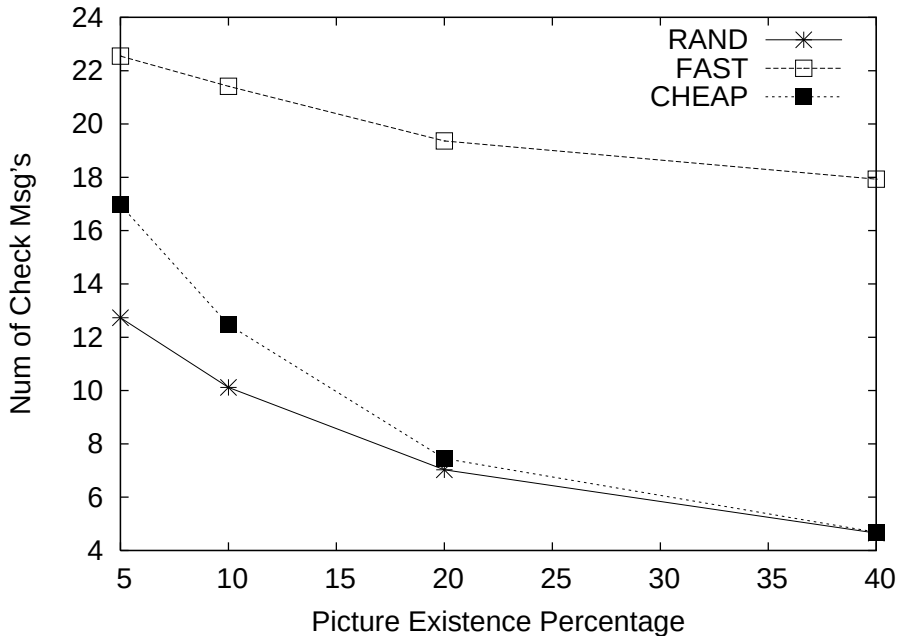

Figure 9.29: A comparison between the average number of check messages as a function percentage in small group

Small Groups

Figure 9.29 shows how the percentage of agents that had a certain picture influences the average number of needed messages to retrieve the picture. As the percentage increases, more agents have the requested picture, and therefore it takes less messages to find an available agent. The FAST algorithm used significantly (paired t-test, $p < 0.001$) more messages, but it is faster and fairer. Up until an availability of 40% the CHEAP algorithm used significantly (paired t-test, $p < 0.03$) more messages than the RAND algorithm, as is the case in the larger groups.

The difference between the influences of availability in small and large groups is due to the fact that in small groups, there is need of fewer messages to find who has a given picture. In contrast, in large groups the influence of one agent on the load of the others is less pronounced, because there are more agents that can query for pictures at parallel. The same reasoning accounts for the differences in retrieval times between large and small groups.

Figures 9.30 and 9.31 show that the RAND and CHEAP algorithms maintain a consistent relationship (compare to Figure 9.29). Surprisingly, the behavior of the FAST algorithm is seemingly in some discrepancy with its behavior in Figure 9.29 (in the large group experiments, there was no discrepancy). The maximum

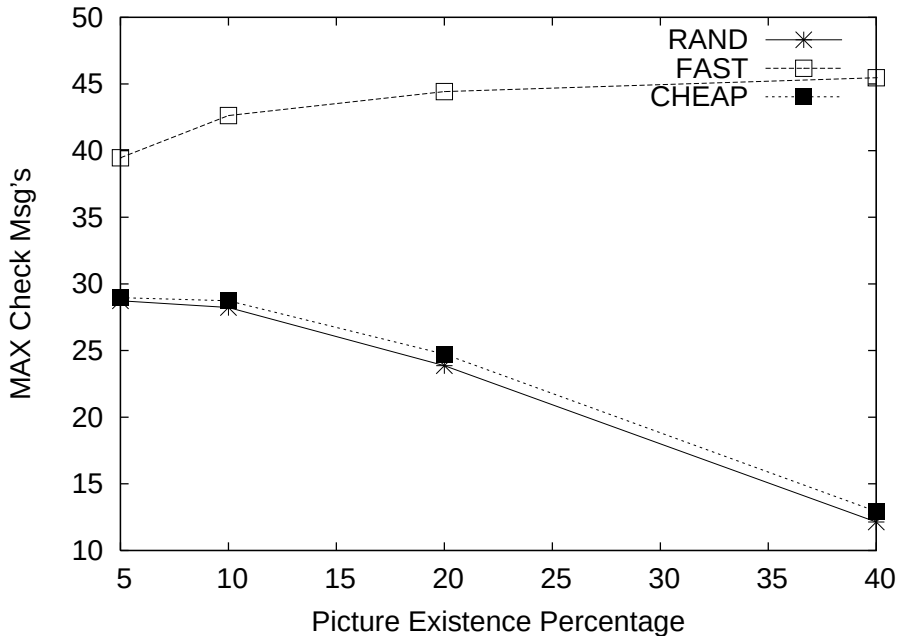

Figure 9.30: A comparison between the average maximum number of check messages as a function of percentage in small group

and the standard deviation rise while the average numbers of messages decrease. The reason for this is that with increased availability, there are more agents to query. For the average case, this helps finding good candidates faster. However, in the worst case, the number of candidates to be contacted is larger.

Figure 9.32 shows that influence of availability on fairness is significant. There is significant difference (paired t-test, $p < 0.0001$) between all the algorithms, where the FAST is fairer than CHEAP and RAND. As availability increases, the number of potential available sources increases, and thus greater opportunity for fair decision. Under percentages of 10%, there is low probability to find agents that have the pictures in a small group, and the RAND algorithm's fairness remains essentially constant. FAST and CHEAP are fairer still, however.

Similar relationship is found for the influence of availability on retrieval time. See Figures 9.33, 9.34 and 9.35. Also contrast with similar findings for the large groups.

9.2.4 Offline Agents

Up to this point, we have dealt with classic closed groups, where all agents are online and available for communications. In this section we explore the performance of the different algorithms when some of the agents are offline, and are not

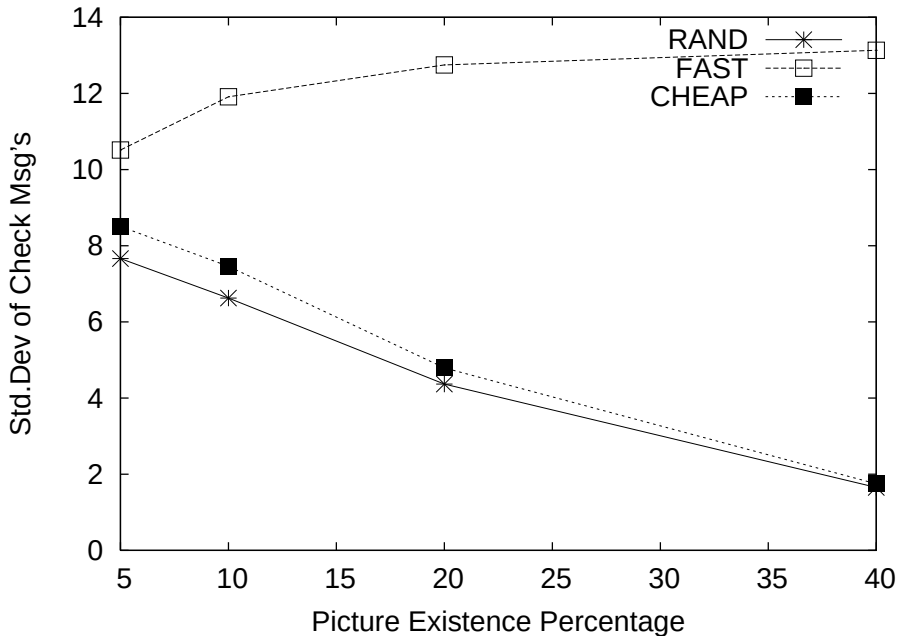

Figure 9.31: A comparison between the average standard deviation of querying as a function of percentage in small group

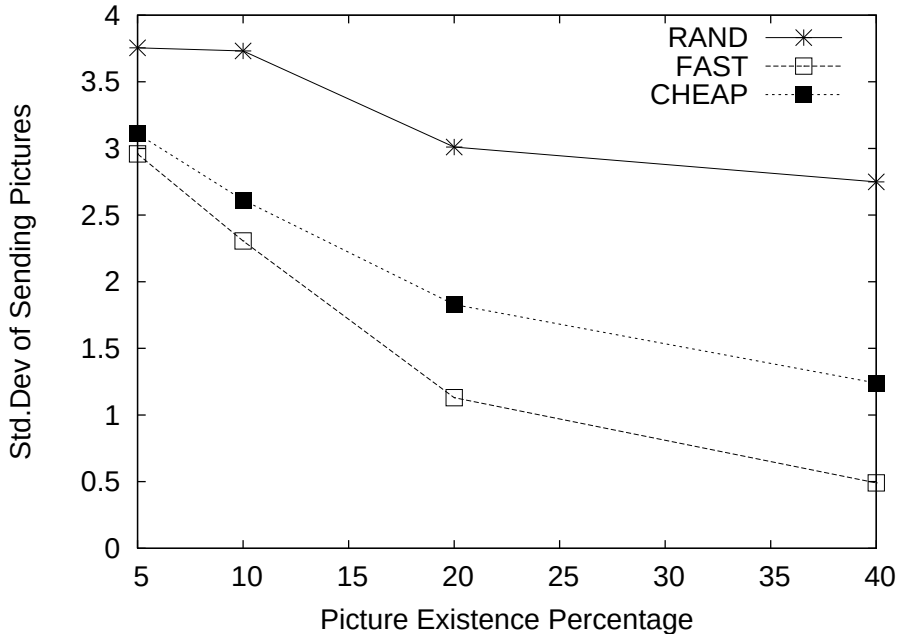

Figure 9.32: A comparison between the average standard deviation of sending Pictures as a function of percentage in small group

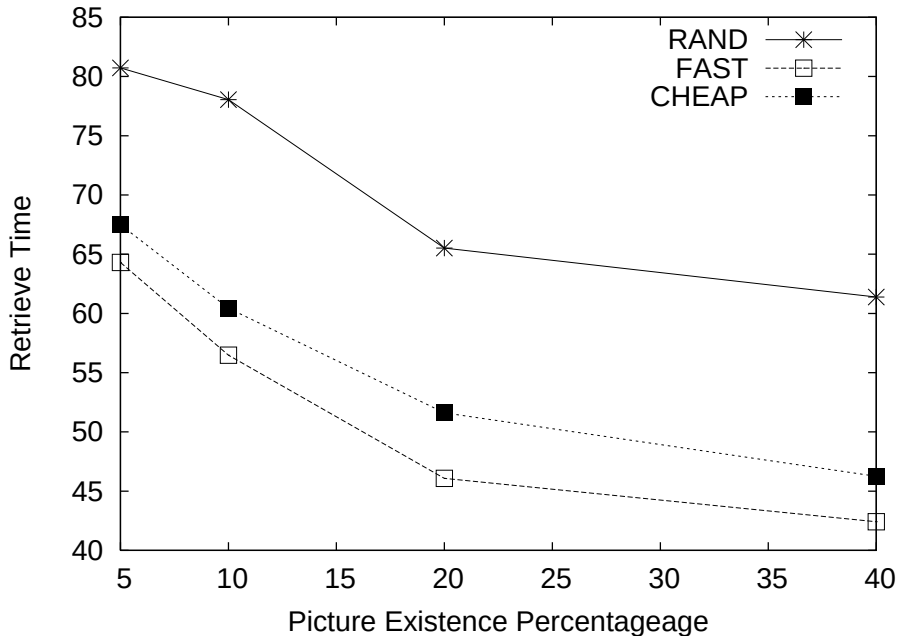

Figure 9.33: A comparison between the average retrieval time as a function of percentage in small group

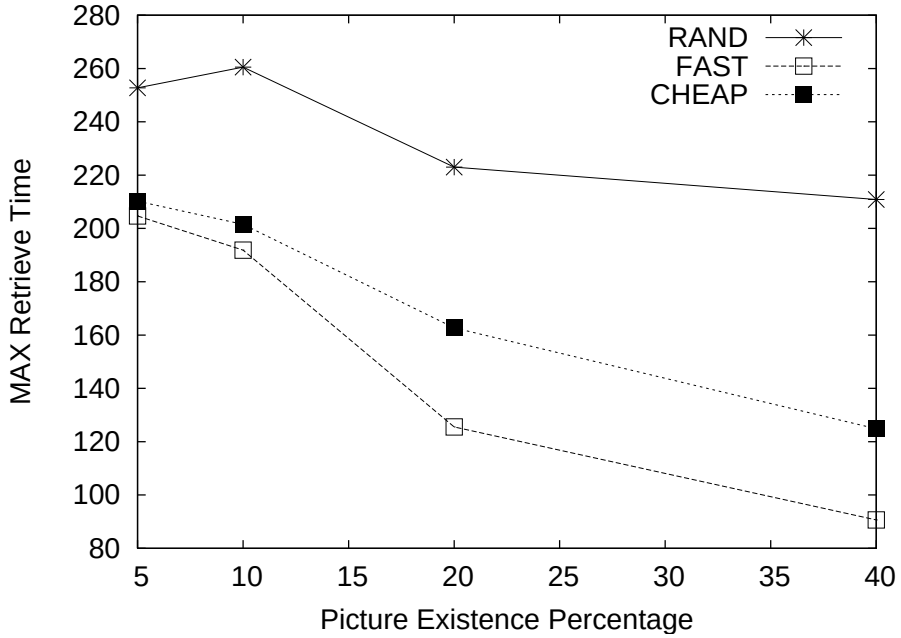

Figure 9.34: A comparison between the average maximum retrieval time as a function of percentage in small group

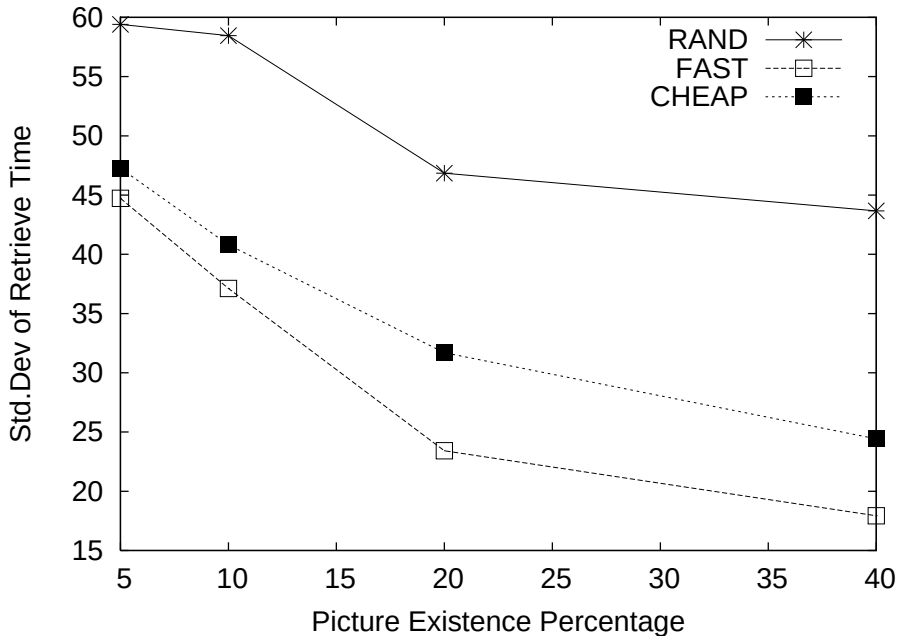

Figure 9.35: A comparison between the average standard deviation of retrieval time as a function of percentage in small group

available (though without the knowledge of their peers).

To do this, we arbitrarily disabled 20% of the agents, such that they were part of the groups, but were offline (could not communicate with their peers). We fixed the size of the pictures to 4MB, the size of the group to 30 agents, and the availability to 20%. This configuration was repeated 30 times. We compared the results to the same group but without offline agents.

	RAND	FAST	CHEAP
Average Std.Dev of Sending Pictures Online	3.06	1.23	1.8
Average Std.Dev of Sending Pictures 20% Offline	3.4	2.02	2.34
Ratio Between Them	0.9	0.6	0.76

Table 9.1: Differences between Average Std.Dev of Sending Pictures

Table 9.1 shows the fairness results for the different algorithms, under both settings. FAST and CHEAP were always fairer. However, the ratio shows that their fairness decreased in the offline settings.

The average retrieval time (Table 9.2) is always faster in our algorithms, con-

	RAND	FAST	CHEAP
Average Retrieve Time Online	66.67	47.38	52.89
Average Retrieve Time 20% Offline	72.57	47.75	52.08
Ratio Between Them	0.91	0.99	1.01

Table 9.2: Differences between Average Retrieve Time

sistently with their being fairer. Surprisingly, it can be seen from the ratio that the retrieval time did not change when less agents could help in each search. The reason for this is that while less agents could respond to queries, less agents sent out queries.

	RAND	FAST	CHEAP
Average Num of Check Msg's Online	6.9	19.56	7.8
Average Num of Check Msg's 20% Offline	8	37.02	13.9
Ratio Between Them	0.86	0.52	0.56

Table 9.3: Differences between Average Num of Check Msg's

The average number of queries sent (Table 9.3) is always higher in our algorithms, as expected. It can be seen from the ratio that the number of queries is influenced drastically when 20% of the agents are offline. Indeed, the number of queries is doubled. Figure 9.3 supports this: It shows that as the group size decreases, the gap between our algorithms to RAND increases. In this case our algorithms detect that less agents are online and behave like a smaller group.

	RAND	FAST	CHEAP
Total Num of unnecessary Check Msg's Online	787.9	1930	877.4
Total Num of unnecessary Check Msg's 20% Offline	712.8	1582.1	811.5
Ratio Between Them	1.1	1.21	1.08

Table 9.4: Differences between Total Num of unnecessary Check Msg's

Finally, in Table 9.4 we measured the messages that were sent to the offline agents, in the two settings. The difference shown is due to two factors: The fact that the offline agents did not send messages to themselves, and the fact that they

did not respond to the online agents' queries. It seems that a pronounced reduction in ratio is similar for all three algorithms.

As can be seen in Table 9.3, our algorithms send more messages (twice) in general, but Table 9.4 shows that they do not send them to the offline agents. So in fact, CHEAP and FAST send more messages but not to the offline agents, i.e., they recognize them as unavailable (though they classify them as too busy, not as offline).

Chapter 10

Discussion and summary

We describe a solution to iterative search problems, using an example of virtual album application. The environment of our problem assumes there are others that will agree to help. And that we will agree to help others. This environment is a classic example for shared activity. We propose a solution for the general problem, and demonstrate it for the virtual album problem. We suggest a solution of distributed decision-making. The decision of an agent on whom to contact is based on the estimated load of others and on the other's willingness to help it. Similarly, an agent decides how to react according to load and willingness to the requester.

We evaluate the behavior of our algorithm in a simulation environment and compare it to the random walk algorithm. Results show an advantage for our algorithm regarding the random walk algorithm in most various environmental settings. These advantages are expressed in: retrieving speed of wanted objects and fairness task's distribution. Our algorithm is always fairer than the random walk algorithm and therefore faster. If the user would try to get the picture fast through our algorithm it would always cost him more messages. If the user wants to retrieve in a cheap way but still to keep on fairness, in most cases it would use less or an equal number of messages regardless of the random walk algorithm.

As future work, we suggest to apply additional methodology of Preliminary work. Preliminary work is a heuristic way to reduce uncertainty. Agents evaluate probability according to the model which according to it it acts, the work that agents will do before they will know the expectations (which information they will want to get or to contribute).

There are a few approaches that agents can adopt for Preliminary work. These approaches are depend on the environment and on the group members commitment. The main idea of this stage is to spread the agents profiles and to collect data on other agents' profiles. Basically, there are only two actions that agents can do to spread their profiles before they know their task:

- Send data on your profile to other agents.
- Ask about missing information on other agents profile.

In any case, there is always an option to combine between those actions. The decision which kind of message to send depends on the environment.

Additional future research extensions concern forecasting of what other users would like to receive in the further. There are two options, send them it before they ask for it or send them it before that they even know about it.

Bibliography

[1] K. Battarbee. Defining co-experience. In *DPPI '03: Proceedings of the 2003 international conference on Designing pleasurable products and interfaces*, pages 109–113, New York, NY, USA, 2003. ACM Press.

[2] D. S. Bernstein, R. Givany, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *MOR: Mathematics of Operations Research*, 27, 2002.

[3] G. de Veciana and X. Y. Fairness. Fairness, incentives and performance in peer-to-peer networks. *Forty-first Annual Allerton Conference on Communication, Control and Computing, Monticello, IL,*, 2003.

[4] V. J. Derlega and J. Grzelak. *Cooperation and Helping Behavior Theoreis and Research*. Academic press, 1982.

[5] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS*, pages 137–144. ACM, 2003.

[6] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *AIJ*, 86(2):269–357, 1996.

[7] M. Hadad, G. Armon-Kest, G. A. Kaminka, and S. Kraus. Supporting collaborative activity. In *AAAI-05*, 2005.

[8] IRST. Personal Experience with Active Cultural Heritage Home Page. http://peach.itc.it/home.html, 2005.

[9] N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):1–46, 1995.

[10] G. A. Kaminka and I. Frenkel. Integration of coordination mechanisms in the BITE multi-robot architecture. pages 2859–2866, 2007.

[11] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems, Lecture Notes in Artificial Intelligence (LNAI-830)*, Amsterdam, The Netherlands, 1992. Springer Verlag.

[12] A. K. H. Leung and Y.-K. Kwok. On topology control of wireless peer-to-peer file sharing networks: Energy efficiency, fairness and incentive. In *WOWMOM*, pages 318–323. IEEE Computer Society, 2005.

[13] H. Levesque, P. Cohen, and J. Nunes. On acting together. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99, Boston, MA, 1990.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. *Proceedings of the 16th International Conference on Supercomputing (ICS-02)*, pages 84–95, June 22–26 2002.

[15] W. Müller and A. Henrich. Fast retrieval of high-dimensional feature vectors in p2p networks using compact peer data summaries. In *MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 79–86, New York, NY, USA, 2003. ACM Press.

[16] R. Nair and M. Tambe. Hybrid bdi-pomdp framework for multiagent teaming. *Journal of AI Research (JAIR)*, 23:367–420, 2005.

[17] K. Noervaag, C. Doulkeridis, and M. Vazirgiannis. Taxonomy caching: A scalable low-cost mechanism for indexing remote contents in peer-to-peer systems. In *ICPS'06: Proceedings of IEEE International Conference on Pervasive Services*, 2006.

[18] O. Stock, C. Rocchi, M. Zancanaro, and T. Kuflik. Discussing groups in a mobile technology environment. In *Proceedings of the Multiusers Intelligent Interactive Interfaces Workshop*, 2005.

[19] S. Talman, M. Hadad, Y. Gal, and S. Kraus. Adapting to agents' personalities in negotiation. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 383–389, New York, NY, USA, 2005. ACM Press.

[20] M. Tambe. Toward flexible teamwork. *Journal of AI Research*, 7:83–124, 1997.

[21] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, June 1993.

[22] R. W. Toseland and R. F. Rivas. *An Introduction to Group Work Practice*. Allyn and Bacon, 2001.

[23] E. Woodrow and W. R. Heinzelman. Spin-it: a data centric routing protocol for image retrieval in wireless networks. In *ICIP (3)*, pages 913–916, 2002.

[24] Y. Yang, R. Dunlap, M. Rexroad, and B. F. Cooper. Performance of full text search in structured and unstructured peer-to-peer systems. *IEEE INFOCOM, Barcelona*, 2006.