Real-Time Context Activity Scheduling For Smart Space

Chia-Yen Shih, Marcus Handte and Pedro José Marrón
Sensor Networks and Pervasive Computing Group,
Institute of Computer Science IV,
Universität Bonn & Fraunhofer IAIS
Bonn, Germany
Email: {shih, handte, pjmarron}@cs.uni-bonn.de

***Abstract*—A task management system that automatically optimizes the schedule of a person by context-aware interleaving of activities can enable persons to be involved more efficiently in daily activities of interests. As a technical basis for such a task management system, we propose the activity broker architecture. The core of this architecture is formed by the activity scheduler that can effectively schedule personal activities in different contexts in a real-time fashion. As a first step towards realizing the overall activity broker architecture, we formally define the *Real-Time Context Activity Scheduling Problem*, and present several solution algorithms. Furthermore, we conduct extensive experiments to evaluate and to compare the performance of the proposed algorithms. The results show that a person can obtain more *productive* schedules with the assistance of the activity scheduler.**

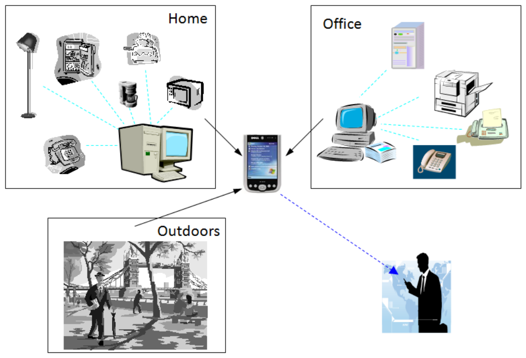


Fig. 1. The Scenario for Context Activity Scheduling.

I. Introduction

Context-awareness is a fundamental element of main-stream productivity-enhancing task management methodologies like GTD [1]. The availability of inexpensive wireless sensors and smart devices enables more and more computer systems to perceive important parts of their environment. This can be - and has been - used to determine different features of the context of a user in performing daily-life activities.

Context information can have a significant impact on personal activities. Given context information about environment dynamics, a person may initiate an activity of interest spontaneously or may involve in more social activities. As a simple example, consider the scenario depicted in Figure 1. Equipped with an adequate positioning system, a smart device can easily determine the location of its user, e.g. whether he is at home, at the office or on-the-move outdoors. Furthermore, by means of networking technology, a smart device can easily discover the resources that are currently available to the user, e.g. whether the user has access to a phone, to a printer, and so on. Clearly, this context can greatly influence the tasks that can be performed.

When performing daily activities, people are often capable of multitasking. The completion of an activity may involve in performing a series of related steps. Switching frequently from on activity to another one in different contexts may raise the complexity of activity completion and result in consequence of performance inefficiency or task incompletion. Thus, an effective activity scheduling mechanism on the basis of the context can help in increasing the user's productivity by organizing real-time activities of various contexts.

In the past, the commercial success of mobile devices has led to the development of several applications for mobile task management. However, current task management systems primarily capture and visualize the tasks that have been defined by the user. Given the inherent context-awareness of efficient task management methodologies, we argue that such context information can be used to automatically optimize task management for the user by scheduling suitable activities.

To enable this, we propose a system architecture called context activity broker. The core of this architecture is formed by the activity scheduler that can effectively schedule the activities of a person in different contexts in a real-time fashion. Our goal is to provide a personal activity broker that can assist the user in taking part in more activities of interests in different contexts, while completing activities with efficient utilization of time. As a first step towards realizing the architecture, we formally define the *Real-Time Context Activity Scheduling Problem*, and present several solution algorithms.

The remainder of this paper is structured as follows. In the next section, we discuss related work. Thereafter, in Section III, we briefly outline the overarching context activity broker architecture. In Section IV, we introduce the *Real-Time Context Activity Scheduling Problem* and we discuss several solution algorithms. In Section V, we evaluate and compare these algorithms and finally, Section VI concludes the paper with a summary and a short outlook on future work.

II. Related Work

The existing work related to real-time context activity scheduling originates from a diverse set of different research areas. The most notable ones are human computer interaction, artificial intelligence and pervasive computing.

In the area of human computer interaction researchers have performed several studies that underline the impact of effective task management on the productivity of persons, e.g. [2]. As a consequence, there has been a long-standing interest on supporting and optimizing task management with electronic tools such as calendars [11], [15], [5] or other productivity software [9], [3] that simplify task execution. However, usually these tools store and visualize the available tasks - possibly with additional information - but they do not try to arrange those tasks by revising optimized schedules. This holds also true for most widely deployed task management systems such as Microsoft Outlook [7] or Mozilla Lightning [8].

In the research area of artificial intelligence, researches have tried to improve task management and execution by providing intelligent agents for personal assistance. CMRadar [12], for example, simplifies calendar management by automating the negotiation of time-slots for meetings. Similarly, PTIME [4] automates the scheduling of meetings but thereby, it also tries to learn user preferences. In contrast to this work which tries to optimize distributed scheduling, the work presented in this paper aims at optimizing a single schedule on the basis of the context of the user. Beyond distributed scheduling, the PExA agent [6] recognizes the activities performed by its user to automate subsequent ones, if possible. Such an automation is orthogonal to the scheduling discussed in this paper.

Finally, in the area of pervasive computing, researchers have pursued a quite diverse set of ideas to improve task management. The Ambush calendar system [14], for example, infers and visualizes the likelihood with which a person attends a particular event scheduled in a calendar. The system presented in [13] tries to identify time slots that are suitable for informal meetings. Although, these systems do not try to actually manage the schedule of a person, their output could be used as an interesting input for context activity scheduling. Probably, the most closely related work is the smart calendar presented in [10]. This calendar suggests possible leisure activities that may fit to the schedule, context and interest of a person. However, the system only considers the spare time of the person and does not try to optimize all tasks.

III. Real-Time Activity Broker Architecture

The primary purpose of the overall architecture is to support the real-time activity broker that assists the user in effectively scheduling activities of various contexts. The broker architecture is capable of considering several important factors that may influence the user's decision when performing daily activities. These factors include *context information*, *preferences*, *activity timing* and *scheduling*. The relation between these factors is illustrated in Fig. 2.

Changing context information is the fundamental element that initiates the activity scheduling. It affects the timing and the user's preferences on activities, and has an impact on scheduling. Based on the analysis of above influential factors, we have designed a broker architecture that assists the user in performing more activities of interests in a smart space. As depicted in Fig. 3, the architecture consists of the following system components:

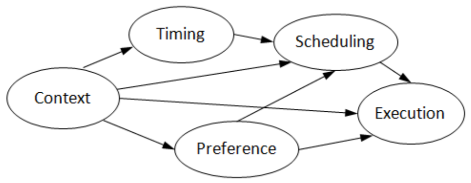


Fig. 2. Influential Factors on Activity Scheduling and Performing

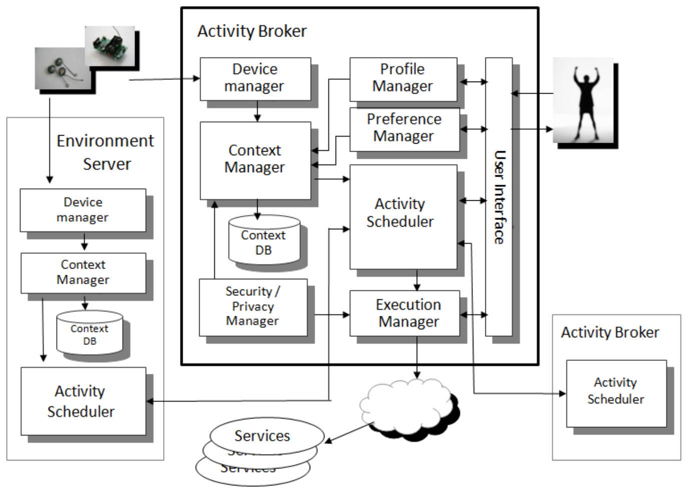
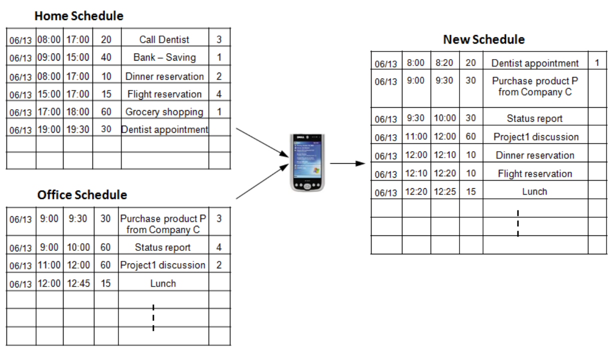


Fig. 3. The Architecture of Our Context Activity Scheduling Broker

- *Device Manager*, which interacts with nearby smart devices, and converts the digital signals into useful context information for the context manager,
- *Context Manager*, which defines and classifies context information that characterizes various aspects of the environment, such as the available objects, present location and other physical phenomena,
- *Profile Manager*, which manages the user's personal data such as the user name, friend/family list and etc,
- *Preference Manager*, which manages the user's preferences in different contexts,
- *Activity Scheduler*, which generates new activity schedules in response to context changes caused by interacting with environment servers, or with other activity brokers for other users,
- *Security/Privacy Manager*, which specifies a set of security/privacy policies to ensure system integrity and to prevent invalid access for disclosing the user's private data, and
- *Execution Manager*, which automatically completes electronic activities, e.g., on-line transactions.

For the remainder of this paper, we focus on the aspect of context-based activity scheduling. Thereby, we provide an in-depth discussion of the core problem — Real-Time Context Activity Scheduling — that the activity broker needs to deal



Home Schedule

06/13	08:00	17:00	20	Call Dentist	3
06/13	09:00	15:00	40	Bank – Saving	1
06/13	08:00	17:00	10	Dinner reservation	2
06/13	15:00	17:00	15	Flight reservation	4
06/13	17:00	18:00	60	Grocery shopping	1
06/13	19:00	19:30	30	Dentist appointment	

Office Schedule

06/13	9:00	9:30	30	Purchase product P from Company C	3
06/13	9:00	10:00	60	Status report	4
06/13	11:00	12:00	60	Project1 discussion	2
06/13	12:00	12:45	15	Lunch	

New Schedule

06/13	8:00	8:20	20	Dentist appointment	1
06/13	9:00	9:30	30	Purchase product P from Company C	
06/13	9:30	10:00	30	Status report	
06/13	11:00	12:00	60	Project1 discussion	
06/13	12:00	12:10	10	Dinner reservation	
06/13	12:10	12:20	10	Flight reservation	
06/13	12:20	12:25	15	Lunch	

Fig. 4. Generating a New Activity Schedule Based on The Current Context

with to achieve effective activity scheduling for the user.

IV. Real-Time Context Activity Scheduling

A. Problem Formalization

Following the previously mentioned methodologies, we assume that a person can only perform one activity at a time and that activities are classified on the basis of their context C, as illustrated in Fig. 4. We define an *activity* as the expenditure of effort over some period of time on a particular part of task. Each activity i is associated with five attributes: (1) the starting time s_i—the time at which i *can* be started earliest, (2) deadline e_i—the time at which i *should* be finished, (3) duration d_i—a non-preemptable period required to perform i, (4) context value c_i, $c_i \in C$,—the context-dependent importance, and (5) the actual scheduled period $\tau_i[\tau_{s_i}, \tau_{e_i}]$ to perform i, where τ_{s_i}/τ_{e_i} refers to the starting/end time of τ_i, $\tau_{e_i} - \tau_{s_i} \geq d_i$ and $\tau_{e_i} \leq e_i$.

An event refers to the start or finish point of an activity, and a schedule is a graph $G = (V, E)$, where V is the set of events and E is a set of activities. Based on our assumption, we define $actNum_A(t)$ to denote the number of activities in a scheduler A at given time t. In addition, there can exist precedence relations among a set of activities. We denote P_A the set of partial precedence relations over a schedule A, and the function $p_A(i, j)$ specifies the precedence relation between activity i and j, where $i, j \in A$. That is, activity i needs to be finished before activity j is performed.

The problem of Real-Time Context Activity Scheduling (RT-CAS) is that, given the context period $\Gamma_c[\Gamma_{c_s}, \Gamma_{c_e}]$, where $\Gamma_{c_s}/\Gamma_{c_e}$ is the start/end time of the context period, a set of context schedule $S_c = (V_c, E_c)$, and an existing schedule $S = (V, E)$, generate a new schedule S' with a set of activities $A \in E_c \cup E$ with

$$Max(\sum_{i}^{|E_c|+|E|} c_i x_i), \quad 0 \leq i \leq |E_c| + |E|$$

$$x_i = \begin{cases} 1, & if\ i\ is\ scheduled \\ 0, & otherwise \end{cases}$$

subject to,

1) $\Gamma_{c_s} \leq \tau_{j_e} \leq \Gamma_{c_e}$, $0 \leq j \leq |A|$,
2) $\forall i, j \in A, p_A(i, j) \in P_c \cup P_s$, and
3) $\forall t \in \Gamma_c, actNum_{S'}(t) \leq 1$.

Condition (1) requires that the scheduler ensures the finish time of each selected activity is earlier than the end of the context period; condition (2) preserves the precedence relations among the selected activities; finally condition (3) specifies that there is only one scheduled activity at any time in the context period. We note that the unselected activities are stored by the scheduler and are potential candidates for the next context scheduling.

B. RT-CAS Algorithms

To solve the RT-CAS problem, we provide a simple but effective solution approach. We describe the procedures of our algorithm by separating them into two parts. The first part focuses on how the broker systematically maintains configurable status of the user's activities of various contexts, while the second part discusses how the broker generates a new schedule in response to context changes by adjusting these status parameters of each activity.

Configurable Parameters For Context Activity Scheduling

To perform activity scheduling, the activity scheduling broker considers three parameters: *relative weights* among contexts that represent the suitability of performing activities that require a certain context in the current context, the user's *preference* and *timing* constraints of every schedulable activity.

The broker first classifies the user's activities based on a set of contexts of the user, $C = \{c_1, ..., c_n\}$, $n \geq 0$. For each context c_i, a set of activities are scheduled with respect to c_i. Let $S_i = \{a_1^i, ..., a_{|S_i|}^i\}$ denote the schedule that includes a set of activities, a_k^i, $1 \leq k \leq |S_i|$, with respect to the context c_i. Each activity a_k^i is associated with a context value v_k^i indicating its *importance* in S_i.

For every context c_i, the broker specifies an array, Ω_{c_i}, of *relative weights* to other contexts c_j, $c_j \in C$, and $\Omega_{c_i} = \{\omega_{c_1}^{c_i}, ..., \omega_{c_n}^{c_i}\}$. The value of $\omega_{c_j}^{c_i}$ is obtained using the following relative weight function:

$$\omega_{c_j}^{c_i} = \omega(c_i, c_j) = \begin{cases} 1, & if\ i = j \\ \omega, & if\ i \neq j, 0 \leq \omega < 1 \end{cases}$$

In addition, the broker also needs to decide the values of *preference* and *timing* parameters for activities in response to context changes. We define two functions, $p(a_k^i, c_j)$ and $t(a_k^i, c_j)$, which return respective values that indicate the user's preference and timing constraints of activity a_k^i with respect to another context c_j, $1 \leq j \leq n$.

Real-Time Context Activity Scheduling

When the broker detects a context change from the context c_i to the context c_j, it follows four steps to generate an activity schedule with respect to c_j. The first step is to determine the context value of every schedulable activity with respect to c_j.

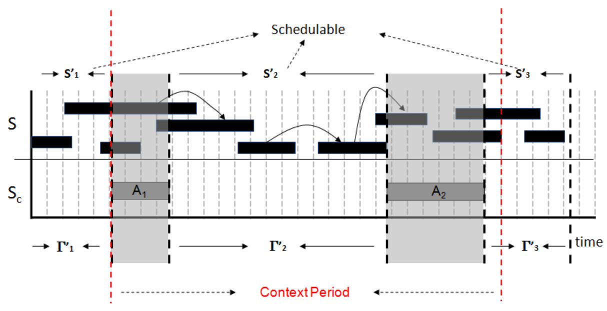
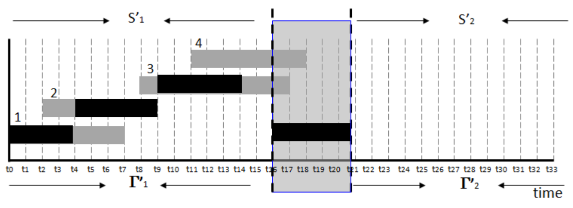


Fig. 5. Basic Idea of Real-time Context Activity Scheduling



Activity i	Context Value c_i	Duration d_i	Start time s_i	Deadline e_i	Scheduled period $\tau[\tau_{si}, \tau_{ei}]$
1	4	4	t_0	t_7	$[t_0, t_4]$
2	3	5	t_2	t_9	$[t_4, t_9]$
3	2	5	t_8	t_{17}	$[t_9, t_{14}]$
4	1	6	t_{11}	t_{18}	*reject*

Fig. 6. Ordered Set of Activities in S Sorted by Highest Context Value First.

This value represents a systematic factor that influences the probability an activity will be selected. The broker computes the context value by manipulating parameter values discussed previously. The set operation $\omega(c_i, c_j)$ is applied to every activity in each context $c_i \in C$ with respect to the context c_j. In other words, the context value of every activity, v^i_k, is multiplied by $\omega^{c_i}_{c_j} = \omega(c_i, c_j)$. The resulted context value is then added to values returned by preference and timing functions, $p(a^i_k, c_j)$ and $t(a^i_k, c_j)$, on the activity with respect to c_j. In summary, the context value of an activity, $v(a^i_k, c_j)$, with respect to the context c_j is expressed as follows:

$$v(a^i_k, c_j) = \omega(c_i, c_j) \cdot v^i_k + p(a^i_k, c_j) + t(a^i_k, c_j).$$

The second step is to identify the schedulable time slots in the given *context period* — the time duration that a set of activities need to be performed based on current context of the user. Fig. 5 shows an example of context period and schedulable time slots: S'_1, S'_2 and S'_3. The gray areas represent non-schedulable time durations that are reserved for performing activities associated with the context c_j (e.g., A_1 and A_2). The black horizontal bars denote the time durations for activities — called *schedulable activities* — that are not associated with c_j, and their context values with respect to c_j are computed by the broker for scheduling.

The third step is to sort all schedulable activities based on their context values with respect to c_j. We call this approach the *Most Context Value First* (MCVF) approach. To explore other possibilities in comparison with the MCVF approach to achieve higher productive scheduling, we also sort the schedulable activities based on different activity properties:

- *Most Value per Duration First* (MVDF): the activities are sorted by the outcome of their context values divided by durations that are required to perform the activities.
- *Short Duration First* (SDF): the activities are sorted by their durations, and the one with the shortest duration is scheduled first.
- *Earliest Deadline First* (EDF): the activities are sorted by their deadlines, and the earliest deadline is listed first for scheduling.

Once all schedulable activities are sorted, the last step for the broker is to select activities to generate the new schedule with respect to c_j. Fig. 6 illustrates a set of activities a_1, a_2, a_3 and a_4 which are candidate activities for the scheduable period S_1. These activities are sorted using the MCVF approach. The table in Fig. 6 shows the sorted order of activities. The broker starts the scheduling by first selecting a_1, specifying its start time, end time, and then continues with the next scheduable activity. The next activity can be scheduled only if its starting time plus its duration is less then the start time of the next schedulable period (e.g., a_2 and a_3). Otherwise, the activity is *rejected* (e.g., a_4) and cannot be scheduled until the next context change.

V. Experiment Evaluation

The purpose of our experiments is to demonstrate that, with the assistance of an activity broker, more activities can be scheduled in response to context changes, and thus can increase the productivity of the user. Thus, we use the number of scheduled activities and the aggregated context value as metrics to evaluate the algorithms.

We conduct three sets of experiments to demonstrate the performance of different scheduling approaches in handling a context change from c' to c. Here we refer context/non-context activities to activities associated with c/c'. For each category, the experiment is setup by first specifying time intervals for both context period P_c and non-context period $P_{c'}$, then two sets of schedules, S_c and $S_{c'}$, are generated for both periods. Finally, we compare performance results that are obtained by averaging outcomes of running experiments on twenty pairs of context/non-context schedules using different approaches. Note that $P_{c'}$ overlaps with P_c, i.e., $P_c \cap P_{c'} = P_c$ and $P_c \cup P_{c'} = P_{c'}$, and we try to schedule the non-context activities in the period $P_{c'}$ into P_c without interfering with the context activities.

In our experiments, the time periods for P_c and $P_{c'}$ are set to be 50 and 150 hours, respectively. To decide the number of activities in a schedule S for a certain period P, we define the formula: $P \geq \rho \cdot (\sum_{i=1}^{|S|} d_i + \sum_{i=1}^{|S|-1} b_i)$, where $|S|$ is the number of activities and $d_i > 0, b_i \geq 0$. P denotes the scheduled period; d_i and b_i denote the duration of activity i, $i \in S$, and the break between activities i and $i+1$, respectively; ρ is a *relaxed coefficient* which indicates the *how busy* the user is. The larger the ρ is, the more relax the user is, and the less

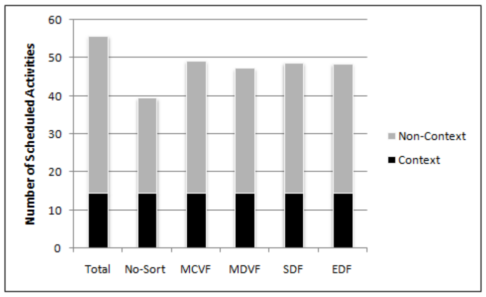


Fig. 7. Total Number Of Scheduled Activities Using Different Algorithms When $\rho = 3$

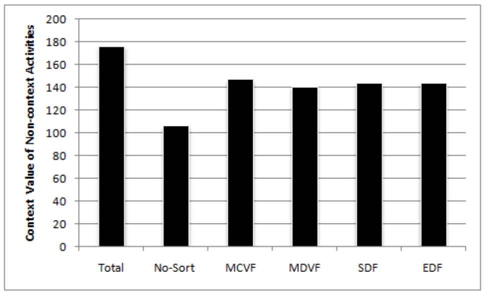


Fig. 8. Total Context Value of Scheduled Non-context Activities Using Different Algorithms When $\rho = 3$

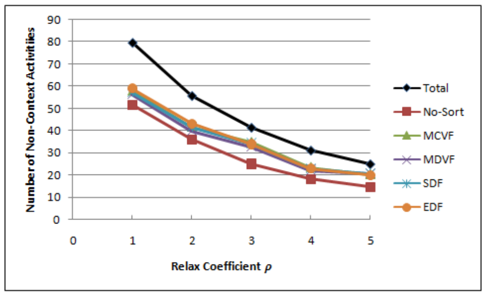


Fig. 9. Number of Scheduled Non-Context Activities With Various ρ — Both Context And Non-context Activities Are Generated Based On The Same ρ

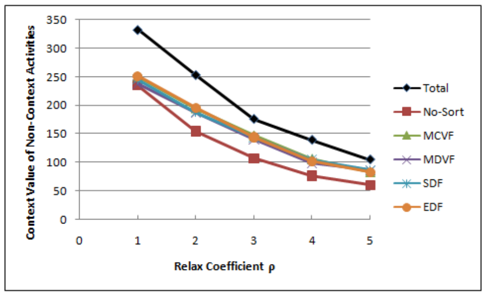


Fig. 10. Total Context Value Of Scheduled Non-Context Activities With Various ρ — Both Context And Non-context Activities Are Generated Based On The Same ρ

activities the user has on the schedule.

The first set of experiments is to provide a basic performance analysis using different scheduling approaches. In these experiments, for each activity i in both S_c and $S_{c'}$, d_i and b_i are generated based on a Poisson distribution. We set $\lambda_d = .75$ hour and $\lambda_b = .50$ hour. In addition, each activity is assigned a context value: the value for a context activity is a fixed number which is set to be 100, while the value for a non-context activity is a uniform random number between 1 and 10. Figure 7 shows the number of scheduled activities using different approaches when $\rho = 3$.

The first bar shows the total number of context/non-context activities generated—some non-context activities are in conflict with context activities in the context period. The second bar indicate the scheduled context/non-context activities without the assistance from the activity broker. In this case, the user only performs context activities in the context period and none of non-context activity is performed. The following bars present the number of context/non-context activities scheduled using MCVF, MVDF, SDF and EDF, respectively. We can see in Figure 7 that all context activities are scheduled in all cases, and more non-context activities are scheduled with the assistance of the scheduler. In general, the number of scheduled non-context activities increases about by 35.75%.

Figure 8 shows corresponding context values for scheduled non-context activities. Recall that a context value represents the *importance* of an activity. The higher the context value is, the more *significant* the activity is and thus should be first considered for scheduling. The result shows an average 35% increase for the aggregated context value with our approaches. Among all approaches, MCVF approach results in the highest aggregated context value. Here we only show the value of non-context activities since all context activities are assigned with a fixed context value.

The purpose of conducting the second set of experiments is to show how the performance changes with different scheduling approaches as the value ρ changes—as the value of ρ increases, the user is more relaxed and thus has less activities for scheduling. In this experiment, activities are created with the same parameter values except for ρ. However, the same value of ρ is used for creating either context or non-context activities. Again, we only show the number of scheduled non-context activities (Figure 9) and their accumulated context values (Figure 10) since all context activities are scheduled and each context activity has a fixed value.

In Figure 9 and 10, the top and the bottom lines represent the total number/value of non-context activities in $P_{c'}$ and the number/value of non-context activities in the non-context period: $P_c \cup P_{c'} - P_c \cap P_{c'} = P_{c'} - P_c$, respectively. The gap of both lines indicates the number/value of non-context activities in the context period P_c. The larger the gap is, the more non-context activities are NOT scheduled in the context period. We can see that all scheduling approaches, represented by lines locating between the top and the bottom lines, result in more scheduled non-context activities with various ρ— about 11.57% more non-context activities can be scheduled using scheduling approaches. When $\rho = 1$, which indicates that the user has more activities in both P_c and $P_{c'}$, there are more context activities in P_c and less schedulable time slots for

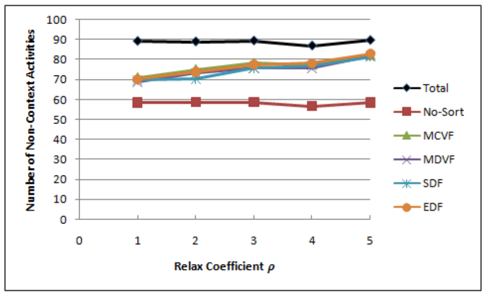


Fig. 11. Number of Scheduled Non-Context Activities With Various ρ — Non-context Activities Are Created With A Fixed $\rho = 2$ While Context Activities Are Generated With Various ρ

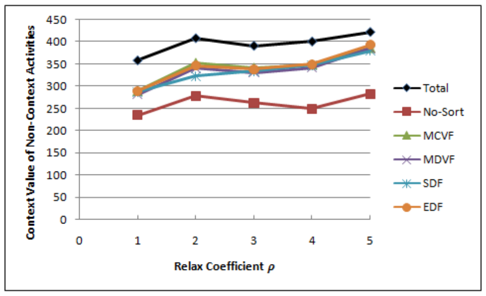


Fig. 12. Total Context Value Of Scheduled Non-Context Activities With Various ρ — Non-context Activities Are Generated With A Fixed $\rho = 2$ While Context Activities Are Created With Various ρ

scheduling non-context activities. Thus, the gap between other lines to the top line is larger. However, with the usage of the scheduler, the gap becomes smaller as ρ becomes larger. This is because more free time improves the effectiveness.

The last set of experiment is to show that when a user has less activities in the context period, the activity scheduler can be better utilized to schedule more non-context activities into the context period. Figure 11 and Figure 12 illustrate how the performance results change when context activities are generated based on various relax coefficient ρ while non-context activities are generated with a fixed value of ρ. Thereby, we create context activities using the same parameter values for the previous experiments; we create non-context activities by setting $\rho = 2$, $\lambda_d = .05$ and $\lambda_b = .25$ hour.

From the figures, we can clearly see that, without using any scheduling algorithm (represented by the bottom line), the difference of number/value of scheduled non-context activities to the total number/value of non-context activities remains almost constant. More non-context activities can be scheduled into the context period using all scheduling approaches. Specially, when the value of ρ increases, all scheduling approaches can achieve higher effectiveness.

VI. Conclusion and Future Work

The context of a person has significant impact on the activities that can be performed. The objective of this paper is to provide a sound real-time context activity scheduling mechanism to assist persons to improve the management of their daily activities. The contribution of this paper is three-fold. We first propose the real-time context activity broker architecture as a basis for context-aware activity scheduling. Then we formally define the Real-Time Context Activity Scheduling (RT-CAS) Problem. Finally, we provide a variety of solution algorithms and we compare their suitability and performance. The experimental results show that a user can obtain more *productive* schedules with the assistance of our activity scheduler. At the present time, we are developing the supportive components of the activity broker architecture. The final result will be an integrated application that can be executed on mobile devices such as smart phones and PDAs.

References

[1] David Allen. *Getting Things Done. The Art of Stress-Free Productivity*. Penguin, 2002.

[2] V. Bellotti, B. Dalal, N. Good, P. Flynn, D. G. Bobrow, and N. Ducheneaut. What a to-do: studies of task management towards the design of a personal task list manager. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 735–742, New York, NY, USA, 2004. ACM.

[3] V. Bellotti, N. Ducheneaut, M. Howard, and Ian I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352, New York, NY, USA, 2003. ACM.

[4] P. Berry, B. Peintner, K. Conley, M. Gervasio, T. Uribe, and N. Yorke-Smith. Deploying a personalized time management agent. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1564–1571, New York, NY, USA, 2006. ACM.

[5] A. E. Blandford and T. R. G.Green. Group and individual time management tools: What you get is not what you need. *Personal Ubiquitous Comput.*, 5(4):213–230, 2001.

[6] H. Bui, F. Cesari, D. Elenius, D. N. Morley, S. Natarajan, S. Saadati, E. Yeh, and N. Yorke-Smith. A context-aware personal desktop assistant. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1679–1680, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[7] Microsoft Cooperation. Microsoft office outlook homepage. http://office.microsoft.com/en-us/outlook/default.aspx, 2008.

[8] Mozilla Foundation. Mozilla lightning project homepage. http://www.mozilla.org/projects/calendar/lightning/, 2008.

[9] W. Geyer, B. Brownholtz, M. Muller, C. Dugan, E. Wilcox, and D. R. Millen. Malibu personal productivity assistant. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 2375–2380, New York, NY, USA, 2007. ACM.

[10] G. Gkekas, A. Kyrikou, and N. Ioannidis. A smart calendar application for mobile environments. In *MobiMedia '07: Proceedings of the 3rd international conference on Mobile multimedia communications*, pages 1–5, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] C. M. Kincaid, P. B. Dupont, and A. R. Kaye. Electronic calendars in the office: an assessment of user needs and current technology. *ACM Trans. Inf. Syst.*, 3(1):89–102, 1985.

[12] P. J. Modi, M. Veloso, S. F. Smith, and J. Oh. Cmradar: A personal assistant agent for calendar management. In *In Agent Oriented Information Systems, (AOIS*, pages 134–148, 2004.

[13] M. Mühlenbrock, O. Brdiczka, D. Snowdon, and J. Meunier. Learning to detect user activity and availability from a variety of sensor data. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, page 13, Washington, DC, USA, 2004. IEEE Computer Society.

[14] E. Mynatt and J. Tullio. Inferring calendar event attendance. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 121–128, New York, NY, USA, 2001. ACM.

[15] S. J. Payne. Understanding calendar use. *Hum.-Comput. Interact.*, 8(2):83–100, 1993.