Learning User's Scheduling Criteria in a Personal Calendar Agent*

Shih-jui Lin and Jane Yung-jen Hsu
Department of Computer Science and Information Engineering
National Taiwan University
1 Sec 4 Roosevelt Road, Taipei, 106
(02) 2363-5336 x 120
{sjlin, yjhsu}@agents.csie.ntu.edu.tw}

Abstract

Calendar scheduling is a necessary but tedious job in daily life. Even with the help of scheduling software, a user has to specify his/her personal scheduling criteria repeatedly in each delegation. This paper presents a software agent that learns a user's scheduling patterns from past experience, and suggests relevant scheduling criteria when the user wants to arrange a new activity. New schemas of scheduling criteria are induced using *decision trees*. To improve the agent's learning performance, an enhanced decision tree algorithm, *HID3*, is proposed. Moreover, the agent observes the calendar of a user to identify inconsistency between his actual behavior and scheduling criteria. The agent can alert user of such inconsistency and suggest updates to his scheduling criteria. Experimental data show that the proposed personal calendar agent can not only learn a user's scheduling criteria with high accuracy, but also keep up with subsequent changes.

1. Introduction

Calendar scheduling is a necessary but tedious job in daily life. When scheduling an activity, one must consider personal restrictions and preferences as well as external factors such as the schedules of the other participants, and any required resources. There have been software developed to help people manage their calendars and schedule their daily activities. For example, Haynes et al. [1] proposed a community of distributed software agents that can communicate with each other by e-mail, and schedule meetings on behalf of their users using a negotiation mechanism. A user simply specifies the criteria of the meeting to be scheduled, then the agents find a time acceptable to all the participants.

While such agents help automate the task of scheduling, a user needs to specify his scheduling criteria explicitly each time he delegates the task to the agents. To further automate the scheduling process, we designed a personal calendar agent that relieves the user of repeated delegation by suggesting scheduling criteria learned from past experience [2].

In what follows, Section 2 first formulates the problem of calendar scheduling in terms of restrictions and preferences, and Section 3 describes the proposed learning approach. The mechanisms for updating and verifying the learned results are presented in Sections 4 and 5. The experimental results are summarized in Section 6, followed by a discussion of related work and the conclusion.

2. Problem Formulation

When a user delegates her agent to schedule her calendar, she has to specify the set of activities together with all relevant scheduling criteria initially. Each activity is specified by five attributes: *activity name*, *participants*, *location*, *required resources*, and *activity duration*. The goal of a personal calendar agent is to find the best time to schedule each activity by learning the general patterns of a user's scheduling criteria and reusing them in future delegations.

Each scheduling criterion is either a *restriction* or a *preference*. The former defines constraints that must hold in the user's calendar, while the latter indicates choices among alternative schedules. There are three kinds of restrictions as follows.

(1) *Time interval restriction* constrains the time for a single activity. For example, taking MRT can not be scheduled at 2 a.m. since the trains are not in service after midnight.
(2) *Precedence restriction* constrains the ordering of two activities. For example, one must get a passport before traveling abroad.
(3) *Time margin restriction* constrains the time margin between two activities. For example, the time margin between two meetings must be greater than the travel time between the two meeting places.

Similarly, there are three types of preferences.

(1) *Time interval preference* models the user's

* This research was sponsored in part by ROC National Science Council under grant No. NSC-88-2213-E-002-007.

preference for the execution time of a single activity, e.g. preferring working in the morning to at night.

(2) *Precedence preference* models the execution priority of two activities, e.g. doing homework before playing a game.

(3) *Time margin preferenc*e indicates preferred time margin between two activities, e.g. arriving at the airport at least one hour before the flight's scheduled departure.

Unlike restrictions, preferences may be violated. For example, suppose that a user prefers "meeting in the *afternoon*". It is acceptable, while not ideal, to schedule a meeting in the morning in order to accommodate all meeting participants.

The personal calendar agent induces the general patterns of a user's scheduling criteria, which are used as suggestions in scheduling new activities. The learned general patterns are represented as *schemas*. A schema describes the conditions of the activities associated with a specific restriction or preference. For example, given a preference schema "dislike meeting with John in the *morning*", the agent will suggest "dislike *morning*" for all meeting in which John participates. When the user wants to arrange a new activity, the agent suggests relevant scheduling criteria according to the learned schemas.

3. Learning Schemas

3.1 Decision Trees

We use a machine learning approach, *decision tree*, to induce schemas. The schemas for the six kinds of scheduling criteria are learned respectively.

The input to a decision tree is an instance, which is described by a set of attributes; the output of the tree is a classification for that instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute [3].

In the decision tree for restriction schemas, the classification is either *allowed*, or *forbidden*; while in the decision tree for preference schemas, the classification can be one of the values **{***highly preferred, preferred, normal, disliked, highly disliked***}**. For time interval schemas (both the restriction and preference schemas), the instances in the decision tree are *activities*. For precedence and time margin schemas, the instances are *activity pairs*, as each schema is associated with two activities.

For example, the decision tree in Figure 1 represents the time interval preference with respect to *morning*. Each path from the tree root to a leaf is a schema. For instance, the rightmost path in the tree is the schema "dislike doing entertainment outdoors in the morning".

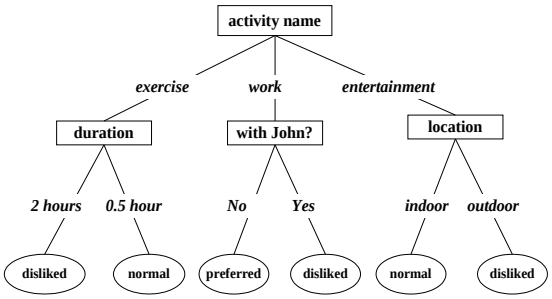

Figure 1: A decision tree representing the time interval preferences with respect to *morning*

Figure 2 shows a decision tree for precedence restrictions. Each precedence restriction is associated with two activities, and indicates whether the execution order "the first activity *before* the second activity" is allowed. Each path in the tree represents a precedence restriction schema. For example, the middle-left path means "meeting before preparing meeting materials is forbidden".

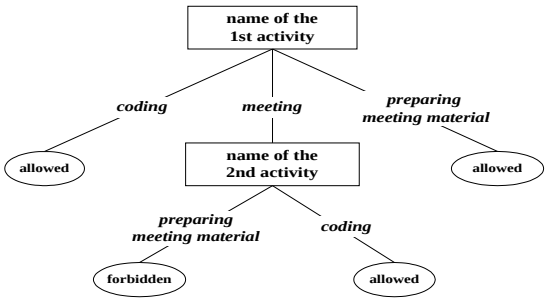

Figure 2: A decision tree representing precedence restrictions

The tree in Figure 3 represents time margin restrictions. It indicates whether it is allowed to execute two activities with a time margin "0.5 hour".

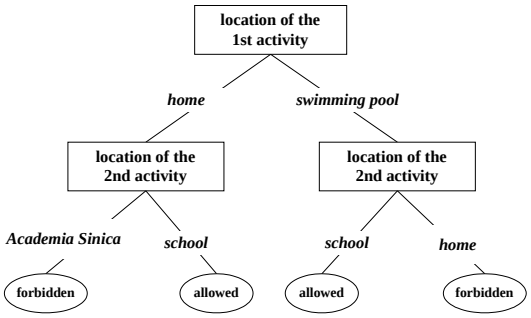

Figure 3: A decision tree representing the restrictions with respect to time margin of *0.5 hour*.

3.2 Hierarchical Concepts

Two of the activity attributes, *activity name* and *location*, may contain *hierarchical concepts*. That is, their values can be grouped into several categories, which can be further grouped into super categories. The resulting categories form a hierarchy of activities.

Take the attribute *activity name* for example. Suppose that there are six values: "swim", "jog", "read", "sing", "study" and "coding". Among these values, "swim" and "jog" belong to "exercise"; "read" and "sing" belong to "enjoyment"; "study" and "coding" belong to "work". Furthermore, "enjoyment" and "exercise" are sub-categories of "leisure". The hierarchy for "activity name" is illustrated in Figure 4.

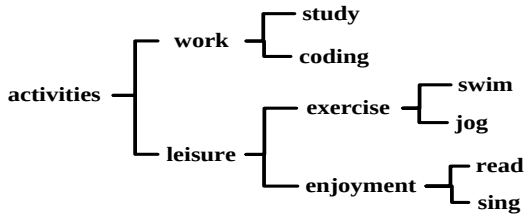

Figure 4: A hierarchy for the attribute "activity name"

A user can categorize and build up such hierarchies, so that activities in the same category have similar scheduling criteria. For example, if a user prefers swimming in the *afternoon*, it is likely that he also prefers jogging in the afternoon, since they both belong to the "exercise" activity. The hierarchy enables the agent to induce more general schemas in terms of categories and to make suggestions for a novel activity in the same category.

3.3 HID3 Algorithm

HID3 (hierarchical ID3) algorithm is designed to improve the performance of decision tree learning by making use of the hierarchical attributes. Similar to ID3, HID3 constructs the decision tree in a top-down fashion, i.e. from root to leaves. At each node, an attribute is selected to classify instances to maximize the information gain. However, when the chosen attribute contains hierarchical concepts, instances are partitioned according to the categories in the hierarchy, instead of their values.

Take the hierarchy in Figure 4 as an example, if the attribute *activity name* is selected for the first time, the decision tree is branched according to the categories on the first level of the hierarchy. That is, the instances are divided into the two categories: "work" and "leisure". When the attribute "activity name" is chosen for the second time, the decision tree is branched according to the categories on the second level of the hierarchy. That is, instances in the "work" category are divided into "study" and "coding", while instances in the "leisure" category are divided into "exercise" and "enjoyment". This process repeats till the entropy of the leaf nodes becomes zero or the lowest level of the hierarchy is reached. Details of the HID3 algorithm are described below.

Algorithm 1 HID3 Algorithm

HID3 (*S, C, A, H*)
Require: A set of instances *S*, a set of attributes *A* describing *S*, a set of classification C for S, and an array of categories *H*, with *H[a]* denoting the current categories for the attribute *a*
Create a node *r* for the tree
if the classification for all instances in *S* is the same **then**
  Return *r* with that classification
end if
if *A* is empty **then**
  Return *r* with the most common classification for the instances in *S*
end if
Choose an attribute *a* from *A*, that best classifies *S*
Assign *a* to the test for *r*
if *a* has hierarchical concepts **then**
  Let *V* be the set of sub-categories of *H[a]*
else
  Let *V* be the set of possible values of *a*
end if
for each $v_i \in V$
  Add a new tree branch below *r*, corresponding to the test $a = v_i$
  Let S_{v_i} be the subset of *S* that have value v_i for *a*
  if S_{v_i} is empty **then**
    Add a leaf node with the most common classification for the instances in S_{v_i}
  else
    if *a* has hierarchical concepts **then**
      if v_i is on the lowest level of the hierarchy of *a* **then**
        Add the sub-tree HID3(S_{v_i} , *C, A-{a}, H*)
      else
        H[a]= v_i
        Add the sub-tree HID3(S_{v_i} , *C, A, H*)
      end if
    else
      Add the sub-tree HID3(S_{v_i} , *C, A-{a}, H*)
    end if
  end if
end for
Return *r*

HID3 has a learning bias: to classify instances with the highest level of categories in the hierarchies.

That is, it tends to generate schemas in terms of more general categories. Therefore, it is able to improve the generalization ability of the learned schemas.

The difference of HID3 and ID3 can be explained from the viewpoint of hypothesis space. When the *activity name* attribute is selected as the classifier, ID3 hypothesizes that "activities with the same name have the same scheduling criteria", while HID3 starts by hypothesizing that "activities in the same category on the first level of the hierarchy have the same scheduling criteria". If the scheduling criteria are different within the same category, the hypothesis of HID3 shrinks and becomes "all activities in the same category on the second level of the hierarchy have the same scheduling criteria". If the scheduling criteria remain different within the category, the hypothesis will keep shrinking and finally become "activities with the same name has the same scheduling criteria", which is exactly the hypothesis of ID3. In other words, HID3 explores more general hypotheses before reaching the hypothesis induced by ID3.

The complexity, in both space and time, of HID is greater than that of ID3. Furthermore, given that decision tree learning performs greedy search without backtracking, the bias of HID3 may result in learning schemas with lower accuracy. The experiment reported in Section 6.1 compares the performance of the two learning algorithms, and examines if HID3 is able to make better predictions when the scheduling criteria involving more hierarchical concepts.

4. Updating Schemas

The user's scheduling criteria may change over time. After schemas are learned, the agent should also keep track of subsequent changes by updating the learned schemas.

To this end, the agent keeps a restriction (or preference) value for each schema. The restriction value *r* is a real number such that

(1) $r = 1$ if the schema indicates that the execution time is allowed.
(2) $r = -1$ if the schema indicates that the execution time is forbidden.

On the other hand, the preference value *p* is any integer from the set {-2, -1, 0, 1, 2}, indicating that the preference is {highly preferred, preferred, normal, disliked, highly disliked} respectively.

The agent captures the changes by continuously updating the restriction and preference values of each schema. Given a restriction schema σ_r, the restriction value *r* of σ_r is updated iteratively each time when the user modifies the restriction, which is suggested according to σ_r:

$$r_{new} = \alpha \times R + (1 - \alpha) \times r_{old}$$

where α is learning rate in [0,1], and *R* is in {1, -1} to indicate if the restriction is {allowed, forbidden} after the user's modification.

Similarly, the function for updating preference value *p* is:

$$p_{new} = \alpha \times P + (1 - \alpha) \times p_{old}$$

where *P* is in {-2, -1, 0, 1, 2} to indicate that the preference is {highly preferred, preferred, normal, disliked, highly disliked} after the user's modification.

By updating the restriction and preference values, the agent is able to adapt itself to the changes, and make updated suggestions for the user's scheduling criteria.

5. Verifying Schemas

Research in cognitive psychology shows the *labile nature* of human's preferences. Preferences specified by a person may be inconsistent: one may prefer an alternative *A* to *B*, prefer *B* to *C*, but prefer *C* to *A* ("intransitivity of preferences" [4]). Even if the preference is consistent, when it is represented in terms of numerical values, these values may be asymmetrical. For example, one may rate the alternative *A* as value "2", when the alternative *B* is taken as the standard (whose value is "0"). On the other hand, when the alternative *A* is taken as the standard, *B* may not be rated as "-2", but "-3" or "-1" ("asymmetry in preferences" [5]). Therefore, modeling preferences by user-input values is straightforward, but may not be able to reflect the preferences accurately. There may be inconsistency between the user's scheduling criteria and scheduling behavior. For example, the user may consider a time interval highly disliked, but always schedule activities in it, even though other time intervals are available.

Since schemas are induced from the scheduling criteria specified by the user, the agent has to verify the schemas by keeping track of the user's scheduling behavior (i.e., the actual calendar). The basic idea is to calculate the frequency of violation of a preference schema. If the frequency is too high, the agent suggests the user to ignore the preference.

Given a preference schema σ_p, the violation frequency *v* of σ_p is updated each time when the activity defined in σ_p is scheduled:

$$v_{new} = \alpha \times V + (1 - \alpha) \times v_{old}$$

where α is learning rate, and V is a Boolean value to indicate if σ_p is violated in the actual calendar.

With this mechanism, the agent is able to help the user clarify what he/she really wants and learn the user's scheduling criteria with better accuracy.

6. Experiments

6.1 Experiments for Learning Schemas

The training instances in the experiments are generated semi-automatically. We collected two users' daily activities, relevant scheduling criteria, as well as the hierarchies for *activity name* and *location*. The users have a total of 20 and 30 scheduling criteria, respectively. We selected 25 activities for learning time interval restrictions and preferences; and 15 activities for learning precedence and time margin restrictions and preferences.

The training instances are generated according to these schemas and activities. Noise is added into the training instances randomly. The number of training instances needed is calculated according to learning theory to get PAC (probably approximately correct) results [6]. There are 196 instances for time interval restrictions; 1583 for time interval preferences; 425 for precedence and time margin restrictions; and 3640 for precedence and time margin preferences.

We experiment both ID3 and HID3 in inducing decision trees. Training instances are divided into five groups, and the learned schemas are evaluated by cross-validation.

The experimental results show that the user's scheduling criteria can be learned by using decision trees. The accuracy is shown in Tables 1 and 2.

Accuracy (%) training/testing	**User A**		**User B**	
	HID3	ID3	HID3	ID3
Time interval	98.5/85.7	98.5/82.7	99.6/91.8	99.6/89.7
Precedence	99.6/98.2	99.6/98.8	99.4/99.4	99.4/99.4
Time margin	100/97.3	100/96.8	99.3/91.8	99.3/87.1

Table 1: The accuracy of learning restriction schemas

Accuracy (%) training/testing	**User A**		**User B**	
	HID3	ID3	HID3	ID3
Time interval	99.4/93.2	99.4/88.4	99.4/98.9	99.4/97.6
Precedence	100/91.7	100/90.5	100/94.4	100/80.9
Time margin	99.9/90.5	99.9/90.4	99.9/96.9	99.9/97.1

Table 2: The accuracy of learning preference schemas

The two algorithms, HID3 and ID3, perform the same in the training set. In the testing set, HID3 has better accuracy then ID3 if there are more hierarchical concepts in the instances (e.g., **User A**'s time interval preferences; **User B**'s time margin restrictions and precedence preferences); the two algorithms have similar performance if the schemas contain few hierarchical concepts. This result shows that using HID3 instead of ID3 will not degrade the performance even if there are few hierarchical concepts in the key schemas.

Consider the computation time of the two algorithms. HID3 needs more time than ID3, but the difference is less than 30%.

In summary, HID3 matches the performance of ID3 with a slight computational overhead. On the other hand, HID3 induces more general schemas and makes better predictions when the learning target contains hierarchical concepts.

6.2 Experiments for Updating Schemas

We randomly generated 1500 schemas for restriction and preference respectively, and simulate the user's changes by modifying each schema arbitrarily.

The results show that changes of restrictions and preferences can be captured by updating the restriction value (and preference value) of schemas. On average, it takes about 2 to 5 iterations to keep up with changes in restrictions; and 3 to 6 iterations to keep up with changes in preferences.

7. Related Work and Conclusion

There are two pieces of important research in this problem domain. One is CAP (Calendar APprentice) by Mitchell et al [7], and the other is the learning interface agent by Maes [8, 9]. Both of them are learning agents that help people in calendar scheduling.

The CAP agent observes the user's scheduling behavior, and helps the user by suggesting meeting parameters (e.g., location, duration, date, time). These suggestions are based on some rules induced by decision trees from the meetings scheduled by the user earlier.

Maes's learning interface agent learns from the user's behavior by memory-based learning, and predicts the user's action in a particular situation. The agent may suggest an action to the user (e.g., to accept an invitation, to reschedule a meeting) by comparing current situation against previous experience.

Both of them focus on learning and predicting the *execution time* of the user's activities. However, suggesting execution time per se is hardly meaningful, especially when the activity involves other participants and/or resources. The user cannot determine the execution time without considering external factors of scheduling (e.g., the preferences of other participants, the schedule of required resources). Therefore, the execution time becomes undecided and unpredictable for any single user, let alone the agent.

It is also problematic to learn from execution time. The execution time is a compromise between the user's preferences and other external factors. The user may violate the personal preferences in order to give way to the scheduling criteria of other participants or resources. However, the user may maintain the same preferences the next time around and *will* arrange activities in the preferred time without conflicts from external factors. In other words, "execution time" is not necessarily the "preferred time". As a result, learning from execution time does not always reflect the actual preferences of the user.

The other problem is, it is too specific to induce preferences from execution time. For example, the user prefers "meeting with John in the afternoon". Suppose the meetings happened to be scheduled on Wed. afternoon for several times. The agent observes the calendar and concludes: "the user prefers Wed. afternoon". Actually, the user has no preference for Wed. over other days. By observing the calendar only, it is impossible and unreasonable for the agent to generalize from "prefer Wed. afternoon" to "prefer afternoon". In other words, the execution time for meetings is too specific to learn, and the learned result is only a small part of the user's preference. The agent cannot get the whole picture of the user's preferences, until all kinds of meetings were scheduled in every possible time slot at least once. That is to say, the agent is not able to predict the restrictions and preferences for a novel <meeting, execution time> pair, due to the lack of generalization ability.

Instead of execution time, the proposed personal calendar agent learns to predict *scheduling criteria*. When the user wants to arrange an activity involving external factors, it is more plausible for the agent to suggest scheduling criteria. Such criteria need to be exchanged with the other participants. An activity is then scheduled based on the scheduling criteria of all participants and resources.

It is also easier to learn from scheduling criteria than execution time. The user writes scheduling criteria for activities, and the agent induces schemas of the scheduling criteria. Because of the generalization power of learning, the agent is able to suggest scheduling criteria for previously seen activities as well as for novel activities. Since the agent also keeps track of the actual calendar and calculates the violation frequency of each schema, it can not only repeat the user's scheduling criteria, but also suggest useful modifications proactively.

This paper has presented an agent to learn a user's scheduling criteria. The experimental evidence shows that the agent can learn a user's scheduling criteria with high accuracy, and keep up with subsequent changes effectively. Such a learning agent can work with other calendar scheduling software to automate the scheduling process and to improve the quality of the resulting schedule.

References

[1] T. Haynes, S. Sen, N. Arora, and R. Nadella. An automated meeting scheduling system that utilizes user preferences. In *Proceedings of Agents '97 Conference*, 1997.

[2] S. J. Lin. Learning restrictions and preferences in a personal agent for calendar scheduling. Master's thesis, National Taiwan University, Taipei, Taiwan, 2000.

[3] T. Mitchell. *Machine Learning*, chapter 3. New York: McGraw-Hill, 1997.

[4] A. Tversky. Intransitivity of preferences. *Psychological Review*, 76:31-48, 1969.

[5] P. D. Tyson. Do your standards make any difference? Asymmetry in reference judgments. *Perceptual and Motor Skills*, 63:1059-1066, 1986.

[6] S. J. Russell. *Artificial intelligence*: A modern approach, chapter 18. London: Prentice Hall International, 1995.

[7] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with learning personal assistant. *Communications of the ACM*, 37:81-91, 1994.

[8] P. Maes and R. Kozierok. Agents that reduce work and information overload. *Communications of the ACM*, 37:31-40, 1994.

[9] R. Kozierok and P. Maes. A learning interface agent for scheduling meetings. In *Proceedings of Intelligent User Interfaces '93*, 1993.