The Discovery of Action Groups for Smart Home

Hsien-Chou Liao[1,*] Bo-Yu Lai[1] Sheng-Che Hsiao[2]

[1]Department of Computer Science and Information Engineering

Chaoyang University of Technology

Wufong Township, Taichung County, Taiwan

hcliao@cyut.edu.tw

[2]Department of Information and Computer Education

National Taiwan Normal University

Taipei City 106, Taiwan

Received 12 August 2006; Revised 20 December 2006 ; Accepted 18 January 2007

Abstract: The concept of a smart home has been discussed in recent years. Its primary purpose is to make life more convenient, safe, and fun in various areas, including home automation, security, entertainment, and so on. In order to automate the interactions between the inhabitants and devices in a home, the prediction of the inhabitant's actions is very important. Current prediction algorithms are usually based on the order of actions to be taken. However, the prediction accuracy of those algorithms is not satisfactory. This is because actions can be separated into a set of groups, and actions in the same group are usually taken in an almost arbitrary order. The set of groups should be discovered before the prediction of actions. In this paper, an action group discovery (AGD) algorithm is proposed. The AGD algorithm is based on the 1-order Markov model and a reverse 1-order Markov model. According to the combination of the two models, a set of group pairs is generated. Then, the action groups are generated by merging these group pairs. A group discovery rate (GDRate) is defined to evaluate the efficiency of the AGD algorithm. Experimental results show that the AGD algorithm can discover most action groups in various situations. The AGD algorithm is thus helpful for predicting actions.

Keywords: action group discovery, action prediction, smart home, Markov model

1 Introduction

The evolution of electronic hardware in size and cost has made available the opportunity for equipping inexpensive devices in homes. Many next-generation information appliances (IAs) are equipped with processors for sophisticated communication and control capabilities among themselves and with the outside world. The primary purpose of these devices and interconnected IAs is to support smart services. One of the important services is home automation, i.e., the automation of interactions between inhabitants and devices.

Home automation mainly relies on two kinds of information: an inhabitant's action and location. The actions are those interactions between an inhabitant and the home appliances and devices. A series of actions forms an action sequence. Some researches have focused on the discovery of significant patterns from an action sequence for home automation. For example, E. O. Heierman and D. J. Cook proposed a novel data-mining algorithm, called Episode Discovery (ED), which discovers behavior patterns within an action sequence in a smart home [1-2]. The authors also proposed a Smart Home Inhabitant Prediction (SHIP) algorithm to predict the most likely next action [3]. SHIP matches the most recent actions with a collected action sequence. In experimental results, the greatest prediction accuracies for two data sets were 33.3 percent and 53.4 percent. Obviously, the prediction accuracy of SHIP is not high enough for use in a smart home.

On the other hand, the Markov model is generally used in the prediction of Web page access [4-5]. In attempting to utilize the Markov model to increase prediction accuracy, we discover what is required to predict actions accurately. Unlike Web page access, in which a user must follow a specific order to access desired Web pages, an inhabitant of a home may take a set of actions in different orders. Assume that there are three actions to be taken: turning on an air conditioner (AirConditionerOn), turning on a desk light (DeskLightOn), and turning on a computer (ComputerOn). The orders of the actions could be (DeskLightOn, ComputerOn, AirConditionerOn), (AirConditionerOn, DeskLightOn, ComputerOn), or (ComputerOn, AirConditionerOn, DeskLightOn). There are

[*] Correspondence author

six possible orders, i.e., permutations, for three actions. Assume they have the same frequency to be taken. When the first of the three actions is taken, the prediction accuracy of the most likely next action is only 50 percent. The more actions there are in a set, the smaller the prediction accuracy will be. Being able to correctly predict the order of actions determines prediction accuracy in a smart home. Therefore, the most important thing before starting to predict actions is the discovery of the actions that may be taken together. These actions that are taken together are defined as an *action group*.

In this paper, we propose an Action Group Discovery (AGD) algorithm. The AGD algorithm is mainly based on a positive and reverse 1-order Markov model constructed from an action sequence. The two models are combined and a set of group pairs is generated from the combined model. Action groups are generated by merging the group pairs. In order to evaluate the efficiency of the AGD algorithm, an action sequence generator (ASG) is designed and implemented. A set of parameters, such as loss rate, interlace count, and so on, are defined to generate possible sequences in the real world. A group discovery rate (*GDRate*) is also defined to measure the efficiency of the AGD algorithm. The experimental results show that the AGD algorithm can discover most of the action groups in various situations.

The rest of the paper is organized as follows. Section 2 reviews works related to smart homes. Section 3 presents our methods, including the positive and reverse 1-order Markov models, and the AGD algorithm. Section 4 explains why and how the set of parameters of the ASG is defined and implemented, such as loss rate, interlace count, and so on. Section 5 describes the experiment design and results. Section 6 gives the conclusion of this paper.

2 Related works

The researches related to smart homes can be classified into the following categories. The first category focuses on network communication and resource management. In a smart home, network communication is concerned with communication among information systems, ubiquitous sensors, home appliances, and so on. For example, Y. J. Lin *et al.* focused on power-line communication and proposed a network infrastructure for smart homes [6]. The network infrastructure uses power-line LANs for building integrated smart homes. All information appliances can be seamlessly interconnected using electric wires. The infrastructure can also guarantee QoS for real-time communication.

On the other hand, the management of resources, such as power, storage, computing resources, and so on, is also an important concern. For example, A. Roy *et al.* proposed a location-awareness resource optimization scheme [7]. The asymptotic equipartition property (AEP) in information theory is used to capture an inhabitant's typical path segments. Reserving resources and activating devices are only along a typical path in order to minimize overall energy consumption.

The second category focuses on system architecture and programming environments. Various system architectures have been proposed for smart homes. For example, R. Kango *et al.* proposed the ELIMA (Environmental Life cycle Information Management and Analysis system) system [8]. ELIMA was proposed to promote smart services based on the connected home appliances. Therefore, a so-called ubiquitous culture can be realized in the future. D. Valtchev and I. Frankov proposed a service gateway architecture for smart homes [9]. This architecture emphasizes the realization of an effective and realistic gateway management system. It is based on the work of the Open Service Gateway Initiative (OSGi). Some researches focused on the programming environment to increase the abstraction level, thus increasing the efficiency of systems development. For example, H. Jahnke and J. Stier focused on the key programming challenges for smart homes, such as customizability and extensibility. They proposed a visual tool, called a microCommander, to rapidly program synergetic devices [10].

The third category focuses on smart services and applications. One important service and application in this category is the care of the elderly. For example, V. Stanford built a pervasive computing environment for the care of the elderly [11]. Three kinds of data are gathered in the environment: health vitals, inputs/outputs, and movement. The data offers efficient care allocation for residents. S. Helal *et al.* also built a smart home infrastructure for the elderly [12]. A precision in-door tracking system based on ultrasonic sensor technology was designed into the infrastructure. Three applications, remote monitoring, attention capture, and indoor blind guidance, were implemented to illustrate that many applications were enabled by the infrastructure.

The last category focuses on location detection and action prediction. They are essential for smart homes. If the most likely next location and action of inhabitants can be predicted precisely, the home can be automated and provide many smart services. In addition to action prediction of MavHome in [1-3], S. P. Rao and D. J. Cook also proposed a simple Markov model [13]. Simple Markov model is an enhancement of Task-based Markov model (TMM). It categorizes actions into abstract tasks and uses this information to make prediction. In the experiment for real data, the prediction accuracy is smaller than 25 percent. The interleaves of actions from different tasks causes that the action sequence is unable to divide into tasks exactly. It emphasizes the importance on discovering of action groups for increasing the prediction accuracy. Besides, N. Noury, G. Virone, and T.

Creuzet proposed a rule-based system to infer the best position inside the health-integrated smart home information system (HIS^2) [14]. The spaces in smart homes are classified into three categories: communication (e.g., entrance), exchange (e.g., corridor), and destination (e.g., bedroom). Three types of sensors can be used to detect an inhabitant in a location. A set of rules based on the status of sensors is used to determine the inhabitant's position. The set of rules are Boolean equations that can easily be implemented into the memory of a low power micro-controller.

Many researches related to smart home are still being undertaken. People spend more time in their homes than in any other places. The ultimate goal of these works is to provide a safe, comfortable environment for people to relax, communicate, or learn.

3 Methods

3.1 Positive and Reverse 1-order Markov Model

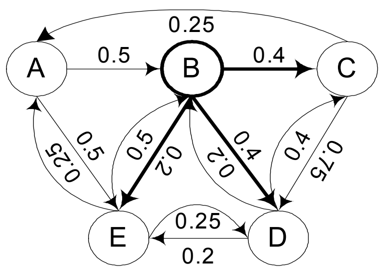
Traditional Markov models are often employed to predict user access behavior. For the prediction of the next Web page a user would most likely access, the traditional Markov model matches a user's current access sequence with the user's historical Web access sequences [4-5]. The order of the Markov model is defined as the length of suffixes of the user's current access sequence for matching. The 0-order Markov model is unconditional probability $p(x_n) = Pr(X_n)$, which is the page access probability. The 1-order Markov model focuses on the transition probability between pages: $p(x_2|x_1) = Pr(X_2 = x_2|X_1 = x_1)$.

The construction of the 1-order Markov model is illustrated in Fig. 1. There are five pages denoted {A, B, C, D, E}. A sample access sequence is listed in Fig. 1 (a). If the possible pages after page B are considered, there are five transitions, which are marked by an underline. Among these five transitions, the transition probabilities of pages C, D, and E are 0.4, 0.2, and 0.4, respectively. All the transition probabilities of the five pages are computed separately. The constructed 1-order Markov model is represented in Fig. 1 (b). An alternative representation of the constructed model is depicted in Fig. 1 (c). In Fig. 1 (c), the 1-order Markov model is represented in a tabular form. The first column shows the current page, and the top row gives the next page. The transition probabilities in Fig. 1 (b) are filled in the corresponding cells in Fig. 1 (c). Therefore, the most likely next page followed by a specific page can be determined from this table. For example, if the current page is D, the next page is most likely C since the corresponding transition probability is higher than that from D to B or D to E.

The 1-order Markov model above is defined as *positive* since the most likely next page is predicted based on the current page. Here, a *reverse* 1-order Markov model is introduced. For the reverse 1-order Markov model, the most likely previous page of the current action is computed based on the same principle but in the reverse direction. For the same example in Fig. 1, the construction of the reverse 1-order Markov model is illustrated in Fig. 2.

{B, C, D, B, D, E, D, C, D, C, A, B, E, A, E, B, E, B, C, D}

(a)



(b)

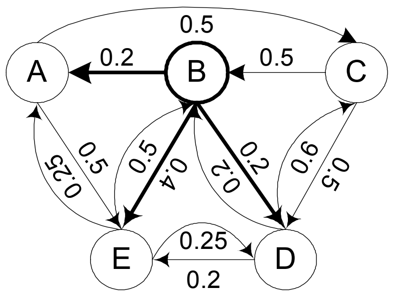
Next / Current	A	B	C	D	E
A	0	0.5(1/2)	0	0	0.5(1/2)
B	0	0	0.4(2/5)	0.2(1/5)	0.4(2/5)
C	0.25(1/4)	0	0	0.75(3/4)	0
D	0	0.2(1/5)	0.4(2/5)	0	0.2(1/5)
E	0.25(1/4)	0.5(2/4)	0	0.25(1/4)	0

(c)

Fig. 1. (a) A sample access sequence (b) 1-order Markov model (c) Alternative representation of the model

{B, C, D, B, D, E, D, C, D, C, A, B, E, A, E, B, E, B, C, D}

(a)



(b)

Previous / Current	A	B	C	D	E
A	0	0	0.5(1/2)	0	0.5(1/2)
B	0.2(1/5)	0	0	0.2(1/5)	0.4(2/5)
C	0	0.5(2/4)	0	0.5(2/4)	0
D	0	0.2(1/5)	0.6(3/5)	0	0.2(1/5)
E	0.25(1/4)	0.5(2/4)	0	0.25(1/4)	0

(c)

Fig. 2. (a) A sample access sequence (b) Reverse 1-order Markov model (c) Alternative representation of the model

According to the constructed reverse 1-order Markov model, the page in front of page B is most likely page E. The positive and reverse 1-order Markov model can be used to predict the most likely page before and after a specific page. The access sequence of web pages is used as an example for illustrating the construction of the reverse 1-order Markov model. However, the reverse model may be meaningless for the prediction of a web page since web pages cannot be browsed in reverse order. On the other hand, the reverse model is mainly used in the discovery of action groups. If the access sequence in Fig. 1 and 2 is the action sequence in a smart home, the reverse model can be used to know the most likely action before a specific action. For a group of actions, an inhabitant usually takes these actions in different orders. The reverse 1-order Markov model becomes useful for action group discovery. For example, in Fig. 1 (c), the next action of action C is most likely D. In Fig. 2 (c), the previous action of action D is most likely C. By examining the action sequence, i.e., access sequence, in Fig. 1 (a), we found that the actions C and D are taken together but in a different order. The basic principle of AGD is based on the combination of the positive and reverse Markov models for discovering possible action groups. The AGD algorithm is presented in the next section.

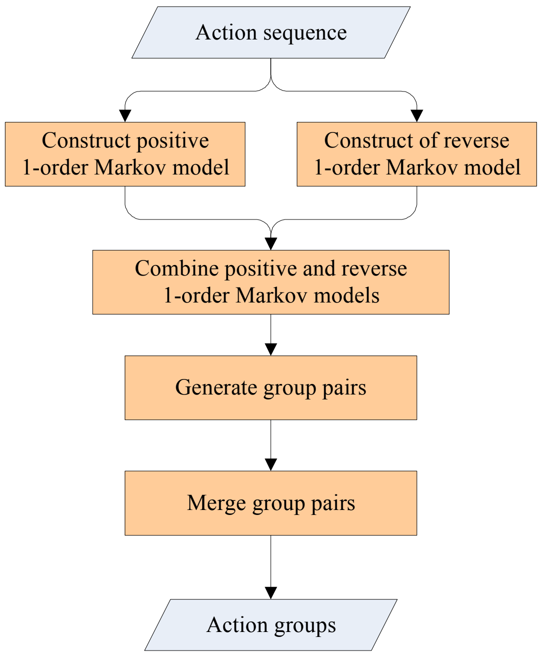

Action sequence
Construct positive 1-order Markov model
Construct of reverse 1-order Markov model
Combine positive and reverse 1-order Markov models
Generate group pairs
Merge group pairs
Action groups

Fig. 3. The process of AGD

3.2 The Principle of AGD

The process of AGD is shown in Fig. 3. The input for AGD is an action sequence. The positive and reserve 1-order Markov models are constructed separately according to the sequence. Then the two models are combined and group pairs are generated from the combined model. Finally, the action groups are obtained by merging the group pairs.

The set of distinct actions in the action sequence is denoted as $A=\{a_1, a_2, a_3, \ldots, a_n\}$. For the positive 1-order Markov model, an action a_i and each action a_j in A is associated with a transition probability $Pr(a_j|a_i)$ [15]. In order to distinguish the probability in the reverse 1-order Markov model, $Pr(a_j|a_i)$ is denoted as $Ppos(a_j|a_i)$. The action a_j and the associated transition probability $Ppos(a_j|a_i)$ are combined into a pair $[a_j, Ppos(a_j|a_i)]$. The n pairs of a_i are collected into a set denoted as Pos_i.

For the reverse 1-order Markov model, the pair of an action a_i is denoted as $[a_j, Prev(a_j|a_i)]$. The n pairs of a_i are collected into a set denoted as Rev_i. The equations for Pos_i and Rev_i are shown as follows:

$$\begin{aligned} Pos_i &= \{[a_1, Ppos(a_1|a_i)], [a_2, Ppos(a_2|a_i)], \ldots, [a_n, Ppos(a_n|a_i)]\} \\ Rev_i &= \{[a_1, Prev(a_1|a_i)], [a_2, Prev(a_2|a_i)], \ldots, [a_n, Prev(a_n|a_i)]\} \end{aligned} \quad \text{-----}(1)$$

The combination of the positive and reverse 1-order Markov models is the combination of Pos_i and Rev_i for each action a_i ($1 \le i \le n$). The probability of $Ppos(a_j|a_i)$ is equal to op_{ij}/t_i, where op_{ij} is the number of times that a_j follows a_i, and t_i is the total number of occurrences of a_i. Similarly, the probability of $Prev(a_j|a_i)$ is equal to or_{ij}/t_i, where or_{ij} is the number of times that a_j is in front of a_i. The equation (1) is rewritten as equation (2).

$$\begin{aligned} Pos_i &= \{(a_1, op_{i1}/t_i), (a_2, op_{i2}/t_i), \ldots, (a_n, op_{in}/t_i)\} \\ Rev_i &= \{(a_1, or_{i1}/t_i), (a_2, or_{i2}/t_i), \ldots, (a_n, or_{in}/t_i)\} \end{aligned} \quad \text{-----}(2)$$

Then the combination of Pos_i and Rev_i, denoted as Cmb_i, is equivalent to the combination of op_{ij}/t_i and or_{ij}/t_i for $1 \le j \le n$. According to the principle of conditional probability, the combination of op_{ij}/t_i and or_{ij}/t_i equals the sum of op_{ij} and or_{ij} divided by two times t_i. The equation is shown below:

$$Cmb_i = \left\{ \begin{array}{l} [a_1, (op_{i1} + or_{i1})/(2 \cdot t_i)], \\ [a_2, (op_{i2} + or_{i2})/(2 \cdot t_i)], \\ \ldots, \\ [a_n, (op_{in} + or_{in})/(2 \cdot t_i)] \end{array} \right\} \quad \text{-----}(3)$$

Cmb_i represents the conditional probabilities that every action a_j occurs either before or after action a_i. After the positive and reverse models are combined, the next step is the discovery of action groups. It consists of two sub-steps. The first sub-step is the generation of group pairs; the second is the merging of group pairs. A *group pair* (a_i, a_j) is a pair where the action a_j is associated with a value higher than others in Cmb_i. The same example shown in Fig. 1 and 2 is used to illustrate the generation of group pairs. The access sequence of web pages is similar to the action sequence in a home. Assume the sequence in Fig. 1 and 2 is an action sequence. The combination of two models in Fig. 1 and 2 is shown in Table 1 (a). In Table 1 (b), the pairs of Pos_3 and Rev_3 come from the tables in Fig. 1 (c) and 2 (c). They are sorted according to the associated probability for the later process. The combination of Pos_3 and Rev_3, i.e., Cmb_3, is sorted, too. The differences of associated probability between two successive pairs are computed and listed in the table. The similar results of Cmb_4 are listed in Table 1 (c). When observing the sample action sequence in Fig. 1, it can be seen that the actions C (a_3) and D (a_4) are taken successively but not in the same order. Therefore, the pair with action D in Table 1 (b) has the higher value among the pairs in Pos_3 and Rev_3, and the pair with action C in Table 1 (c) has the highest value among the pairs the positive and reverse models in Pos_4 and Rev_4. This means that the actions could be in the same group. The combination of the two models causes the pair (D, 0.625) in Cmb_3 and (C, 0.5) in Cmb_4 to have values higher than the others.

Table 1. (a) The combined example of

Actions (a_i)	Cmb_i				
A (a_1)	{(A,0), (B,1/4),	(C,1/4),	(D,0),	(E, 2/4)}	
B (a_2)	{(A,1/10),	(B,0),	(C,2/10),	(D,2/10),	(E, 4/10)}
C (a_3)	{(A,1/8),	(B,2/8),	(C,0),	(D,5/8),	(E, 0)}
D (a_4)	{(A,0), (B,2/10),	(C,5/10),	(D,0),	(E, 2/10)}	
E (a_5)	{(A,2/8),	(B,4/8),	(C,0),	(D,2/8),	(E, 0)}

(b) The differences of pairs in Cmb$_3$

Rank	Pos_3	Rev_3	Cmb_3	Diff.
1	**(D, 0.75)**	(B, 0.5)	**(D, 0.625)**	**0.375**
2	(A, 0.25)	(D, 0.5)	(B, 0.25)	0.125
3	(B, 0)	(A, 0)	(A, 0.125)	0.125
4	(C, 0)	(C, 0)	(A, 0)	0
5	(E, 0)	(E, 0)	(D, 0)	-

(c) The differences of pairs in Cmb$_4$

Rank	Pos_4	Rev_4	Cmb_4	Diff.
1	(C, 0.4)	**(C, 0.6)**	**(C, 0.5)**	**0.3**
2	(B, 0.2)	(B, 0.2)	(B, 0.2)	0
3	(E, 0.2)	(E, 0.2)	(E, 0.2)	0.2
4	(A, 0)	(A, 0)	(A, 0)	0
5	(D, 0)	(D, 0)	(D, 0)	-

Although the combination causes those pairs whose actions may be in the same group to be ranked in front of the sorted list of Cmb_i, the next step is to determine how many pairs could be included in the group pairs. Here, a *threshold* is defined for this purpose. By observing the sorted list of Cmb_i in Table 1 (b) and (c), there is a large difference between the associated values of the pairs that could be included in the group pairs and the others. Therefore, the threshold is defined as the largest value among the differences of two successive pairs in Cmb_i. In Table 1 (b) and (c), the threshold of Cmb_3 and Cmb_4 is 0.375 and 0.3, respectively. Those actions of the pairs in front of the threshold are included in the group pairs. Thus, the group pairs, (C, D) and (D, C), are generated from Cmb_3 and Cmb_4, respectively.

After group pairs of all the Cmb_i are generated, the action group can be discovered according to the following algorithm:

```
Algorithm Merge_Group_Pairs(GP)
Input: GP (a set of generated group pairs)

0Output: AG (a superset of action groups)
begin
    $AG := \varnothing$ ;
    repeat
        get one group pair ($a_i$, $a_J$) from GP
        if $a_i$ or $a_J \in$ an action group ag in AG then
            $ag := ag \cup \{a_i, a_j\}$ ;
        else
            $AG := AG \cup \{\{a_i, a_j\}\}$ ;
        remove ($a_i$, $a_J$) from GP
    until $GP = \varnothing$
end
```

Fig. 4. The algorithm for discovering action groups by merging group pairs

Table 3. An example of discovering action groups from group pairs

Iteration	*GP*	*AG*
0	$\{(\boldsymbol{a_1, a_3}), (a_2, a_4), (a_3, a_1), (a_4, a_2), (a_4, a_5)\}$	$\varnothing$
1	$\{(\boldsymbol{a_2, a_4}), (a_3, a_1), (a_4, a_2), (a_4, a_5)\}$	$\{\{a_1, a_3\}\}$
2	$\{(\boldsymbol{a_3, a_1}), (a_4, a_2), (a_4, a_5)\}$	$\{\{a_1, a_3\},\{a_2, a_4\}\}$
3	$\{(\boldsymbol{a_4, a_2}), (a_4, a_5)\}$	$\{\{a_1, a_3\},\{a_2, a_4\}\}$
4	$\{(\boldsymbol{a_4, a_5})\}$	$\{\{a_1, a_3\},\{a_2, a_4\}\}$
5	$\varnothing$	$\{\{a_1, a_3\},\{a_2, a_4, a_5\}\}$

GP is a set of generated group pairs. An action group is a set of actions. *AG* is the superset of the entire action group. It is initially assigned to an empty set. For each group pair in *GP*, the two actions making up the pair are checked to determine whether one of the actions is included in any action group in *AG*. If it is true, the other action is added to the same action group. Otherwise, a new action group with the two actions is added to *AG*. After all the group pairs in *GP* are checked, the final *AG* contains the entire action groups discovered. Table 3 shows an example that illustrates the above algorithm.

In the above example, although action a_2 does not have a group pair to associate with a_5 directly, a_2 and a_5 are still included in the same action group via another action a_4. This means that AGD can discover an action group without any pair of actions being included in the group pair.

A more complex example used to illustrate the process of AGD is shown in Fig. 5. Fig. 5 (a) shows a partial positive 1-order Markov model constructed from an action sequence. Only op_{ij} of the transition probability is marked on the link for better readability. Fig. 5 (b) shows a partial reverse model. Similarly, only or_{ij} is marked on the link. Fig. 5 (c) shows the combination of the two models. The labels on the links are the summation of the labels on the corresponding links of the two models. The group pairs are marked by bold links, and the discovered action group is marked by the gray area, as shown in Fig. 5 (d).

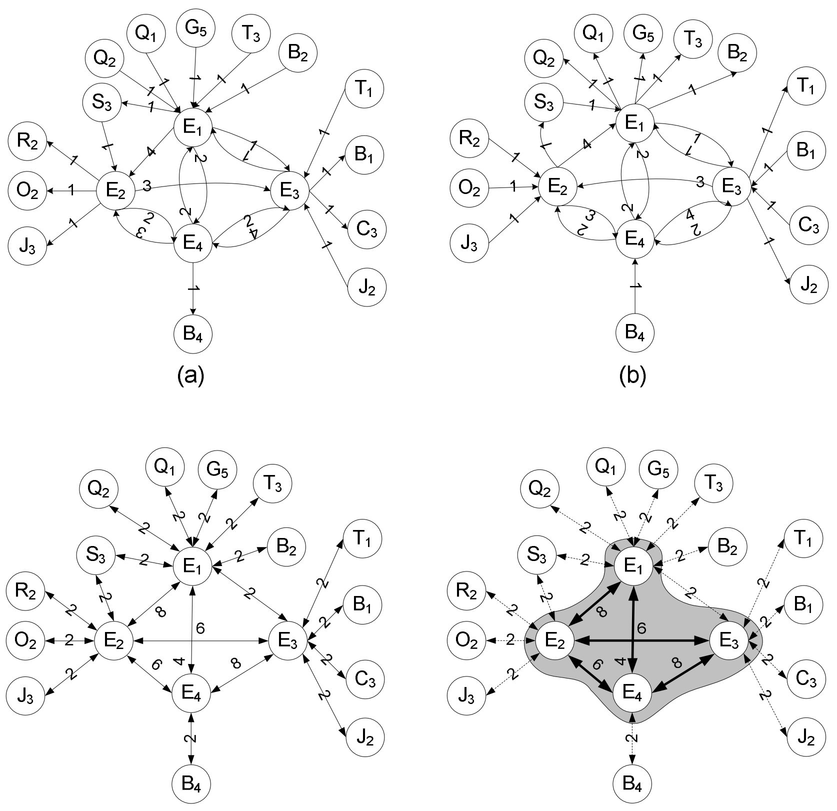


Fig. 5. An example of action group discovery

(a) a partial positive 1-order Markov model (b) a partial reverse 1-order Markov model

(c) a combination of the two models (d) the discovered action group is marked by the gray area

3 Action Sequence Generator

In order to evaluate the performance of AGD, an action sequence generator (ASG) is used. The purpose of an ASG is to provide parameters for generating diverse action sequences. The parameters come from the following observations of actions taken in real life. First, there are different lengths of action sequences, sizes of action groups, and sizes of actions per group. Second, people may take the actions of a group in any order, and partial actions of a group may not be taken. Finally, people may take the actions of another group before finishing the actions of current group. Therefore, partial actions of two groups may be interlaced on the boundary of two groups when people switch from one group to another.

According to the above observations, the parameters of ASG are listed as follows:

- Group size: the size of possible action groups
- Actions per group: the size of actions in a group.
- Total actions: the total actions to be generated in an action sequence.
- Loss rate: a value from zero to 1 to indicate the maximum ratio of actions that do not have to be taken, i.e., loss in the actions taken.
- Interlace count: the maximum number of actions to be interlaced with the actions of the next group.

Table 4. The examples of loss rate and interlace count

(a) The possible results with different loss rate

Action group={a_1, a_2, a_3, a_4, a_5}		
Action orders	Loss rate	Results
a_3, a_2, a_4, a_1, a_5	0	a_3, a_2, a_4, a_1, a_5
a_4, a_5, a_3, a_1, a_2	0.2	a_4, a_5, a_3, a_1, a_2 a_4, a_5, a_3, a_1
a_2, a_4, a_5, a_1, a_3	0.4	a_2, a_4, a_5, a_1, a_3 a_2, a_4, a_5, a_1 a_2, a_4, a_5

(b) The possible results with different interlace count

Action group A	Action group B	Interlace count	Results
a_3, a_2, a_4, a_1, a_5	b_2, b_4, b_1, b_3	0	a_3, a_2, a_4, a_1, a_5, b_2, b_4, b_1, b_3
a_4, a_5, a_3, a_1, $\boldsymbol{a_2}$	$\boldsymbol{b_3}$, b_2, b_4, b_1	1	a_3, a_2, a_4, a_1, a_5, b_2, b_4, b_1, b_3 a_4, a_5, a_3, a_1, $\boldsymbol{b_3}$, $\boldsymbol{a_2}$, b_2, b_4, b_1
a_2, a_4, a_5, $\boldsymbol{a_1}$, $\boldsymbol{a_3}$	$\boldsymbol{b_4}$, $\boldsymbol{b_3}$, b_1, b_2	2	a_3, a_2, a_4, a_1, a_5, b_2, b_4, b_1, b_3 a_4, a_5, a_3, a_1, $\boldsymbol{b_3}$, $\boldsymbol{a_2}$, b_2, b_4, b_1 a_2, a_4, a_5, $\boldsymbol{b_4}$, $\boldsymbol{a_1}$, $\boldsymbol{b_3}$, $\boldsymbol{a_3}$, b_1, b_2

(c) The mixture of loss rate (0.4) and interlace count (2)

Possible results of action group B Possible results of action group A	b_4, b_3, b_1, b_2	b_4, b_3, b_1
a_2, a_4, a_5, a_1, a_3	a_2, a_4, a_5, a_1, a_3, b_4, b_3, b_1, b_2 a_2, a_4, a_5, a_1, $\boldsymbol{b_4}$, $\boldsymbol{a_3}$, b_3, b_1, b_2 a_2, a_4, a_5, $\boldsymbol{b_4}$, $\boldsymbol{a_1}$, $\boldsymbol{b_3}$, $\boldsymbol{a_3}$, b_1, b_2	a_2, a_4, a_5, a_1, a_3, b_4, b_3, b_1 a_2, a_4, a_5, a_1, $\boldsymbol{b_4}$, $\boldsymbol{a_3}$, b_3, b_1 a_2, a_4, a_5, $\boldsymbol{b_4}$, $\boldsymbol{a_1}$, $\boldsymbol{b_3}$, $\boldsymbol{a_3}$, b_1
a_2, a_4, a_5, a_1	a_2, a_4, a_5, a_1, b_4, b_3, b_1, b_2 a_2, a_4, a_5, $\boldsymbol{b_4}$, $\boldsymbol{a_1}$, b_3, b_1, b_2	a_2, a_4, a_5, a_1, b_4, b_3, b_1 a_2, a_4, a_5, $\boldsymbol{b_4}$, $\boldsymbol{a_1}$, b_3, b_1
a_2, a_4, a_5	a_2, a_4, a_5, b_4, b_3, b_1, b_2	a_2, a_4, a_5, b_4, b_3, b_1

Here, three examples are used to illustrate the loss rate and interlace count. In Table 4 (a), the action group consists of five actions. Three orders are associated with different loss rates. When the loss rate equals 0.2, the maximum number of actions that may not be taken is one (5 actions × 0.2 = 1). Therefore, there are two possible results. Similarly, there are three possible results when the loss rate is 0.4. In Table 4 (b), it is assumed that action group A is followed by group B at a specific time. When the interlace count equals zero, the result is the concatenation of groups A and B without any interlacing. When the interlace count is one, it means that at most one action can be interlaced with the actions of the next group. Thus, there are two possible results. In Table 4 (c), the loss rate and interlace count equals 0.4 and 2, respectively. The first column and the first row are the possible results of action groups A and B with the loss rate 0.4. The final results of the six different combinations are shown in the corresponding column and row. If the number of group A's loss actions are greater than or equal to the interlace count, there is no interlacing in the final result. Such a situation can be found in the last row.

The ASG algorithm for generating an action sequence, according to the above parameters, is shown in Fig. 6. In the ASG algorithm, some actions at the tail end of the current group must interlace with the actions of the next group. However, the next group is unknown in the current iteration. The sequence *pre_ia* and a variable *pre_ic* are used to store these actions in the next iteration.

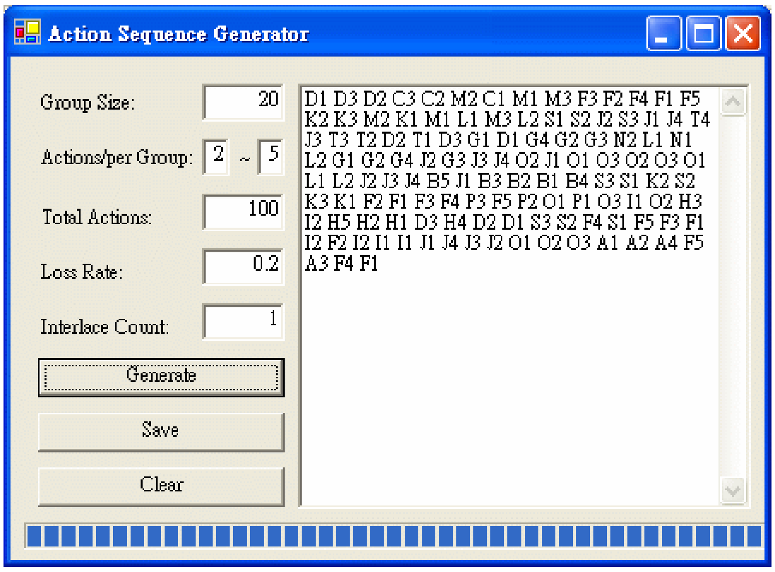
In the ASG algorithm, some actions at the tail end of the current group must interlace with the actions of the next group. However, the next group is unknown in the current iteration. The sequence *pre_ia* and a variable *pre_ic* are used to store these actions in the next iteration.

Algorithm Generate_Action_Sequence(gs, apg_{min}, apg_{max}, ta, loss, ic)

```
Input: gs (group size),
    apg_min, apg_max (the min and max values of actions per group),
    ta (total actions),
    loss (loss rate), and
    ic (interlace count)

Output: AS (the action sequence)
begin
    AS := <>; {empty sequence}
    determine the number of actions for each group by choosing between
        apg_min and apg_max randomly
    pre_ic := 0 ; {the designated interlace count of the previous group}
    pre_ia := <> ; {the actions sequence of the previous group to be interlaced}
    repeat
        choose a group g randomly
        p := a random permutation of the actions in g ;
        n := loss × number of actions in g ;
        choose a number n' between 0 and ⌊n⌋ ;
        remove n' actions from the tail of p
        if pre_ic > 0 then
            p := merge pre_ia and p by interlacing actions in pre_ia and
                the first pre_ic actions in p
        if (ic– n') ≤ 0 then
            pre_ic := 0 ;
        else
            pre_ic := ic – n' ;
        pre_ia := pre_ic actions from the tail of p
        remove pre_ic actions from the tail of p
        AS := AS ^ p ; {concatenation of AS and p}
    until size(AS) ≥ ta
end
```

Fig. 6. The ASG's algorithm for generating action sequence



Group Size: 20
Actions/per Group: 2 ~ 5
Total Actions: 100
Loss Rate: 0.2
Interlace Count: 1

```
D1 D3 D2 C3 C2 M2 C1 M1 M3 F3 F2 F4 F1 F5
K2 K3 M2 K1 M1 L1 M3 L2 S1 S2 J2 S3 J1 J4 T4
J3 T3 T2 D2 T1 D3 G1 D1 G4 G2 G3 N2 L1 N1
L2 G1 G2 G4 J2 G3 J3 J4 O2 J1 O1 O3 O2 O3 O1
L1 L2 J2 J3 J4 B5 J1 B3 B2 B1 B4 S3 S1 K2 S2
K3 K1 F2 F1 F3 F4 P3 F5 P2 O1 P1 O3 I1 O2 H3
I2 H5 H2 H1 D3 H4 D2 D1 S3 S2 F4 S1 F5 F3 F1
I2 F2 I2 I1 I1 J1 J4 J3 J2 O1 O2 O3 A1 A2 A4 F5
A3 F4 F1
```

Fig. 7. A screen shot of the action sequence generator

ASG is implemented by using VB.Net, according to the above algorithm. A screen shot is shown in Fig. 7. All the parameters are listed on the left-hand side. After their values are entered, users can click the "Generate" button to generate an action sequence. The generated sequence is shown on the right-hand side. Users can click the "Save" button for later analysis. A "Clear" button can be used to clear the current sequence for generating a new one. Practically, a behavior watcher or tracer must be installed to collect the real actions that are invoked in a home environment. The real action may include the related information, such as device identifier, location, state, happening time, *etc*. Although the happening time or location is helpful for the action group discovery, the ASG only focus on the device identifier. The AGD algorithm is mainly designed to discover action groups from the sequence of device identifier.

4 Experimental Study

In order to evaluate the effectiveness of the AGD algorithm on action group discovery, an experiment was designed. A group discovery rate (*GDRate*) was defined to represent the effectiveness of the AGD algorithm. The *GDRate* is the percentage of action groups discovered correctly. Its formula is shown below:

$$GDRate = \frac{\text{number of action groups discovered correctly}}{\text{total number of action groups}} \times 100\% \quad (4)$$

Five experiments were designed to measure the *GDRate* of the AGD algorithm influenced by the following parameters below:

- Total actions vs. *GDRate*
- Loss rate vs. *GDRate*
- Interlace count vs. *GDRate*
- Actions per group vs. *GDRate*
- Group size vs. *GDRate*

The parameter settings of the five experiments are listed in Table 5. These settings were based on an inhabitant's possible behavior in a home. For example, the range of actions per group is from two to five actions. These are reasonable values based on our observation in real life.

For each experiment, ten different action sequences were generated for each combination of parameter values. The final *GDRate* was the average of the group discovery rate of the ten sequences. The experimental results are shown in the following figures.

The first experimental result is shown in Fig. 8. The x-axis represents the total number of actions from 100 to 1,000. The y-axis represents the *GDRate*. Four different curves represent the results with the loss rates 0, 0.2, 0.4, and 0.6. According to the figure, the *GDRate* approaches 90% when the total number of actions is greater than or equal to 700. In addition, there is no obvious difference with the *GDRates* for four different loss rates. Although partial actions of a group are lost in the action sequence, the AGD algorithm can still discover most of the action groups successfully.

The second experimental result is shown in Fig. 9. The x-axis represents the loss rates from zero to 0.9. The y-axis represents the GDRate. Three different curves represent the results with the total number of actions of 300, 500, and 700. Theoretically, the increase of loss rate should cause the GDRate to decrease. However, this does not occur, according to the figure. This means that the AGD algorithm does not suffer from an increase of loss rate.

Table 5. The parameter settings of five experiments

Parameters / Experiments	Total Actions	Loss Rate	Interlace Count	Actions per Group	Group Size
A. Total actions vs. *GDRate*	100, 200, …, 1000	0, 0.2, 0.4, 0.6	0	2-5	20
B. Loss rate vs. *GDRate*	300, 500, 700	0, 0.1, …, 0.9	0	2-5	20
C. Interlace count vs. *GDRate*	300, 500, 700	0	0, 1, 2, 3, 4, 5	5	20
D. Actions per group vs. *GDRate*	100, 200, …, 1000	0	0	2, 3, 4, 5	20
E. Group size vs. *GDRate*	100, 200, …, 1000	0	0	2-5	5, 10, 15, 20

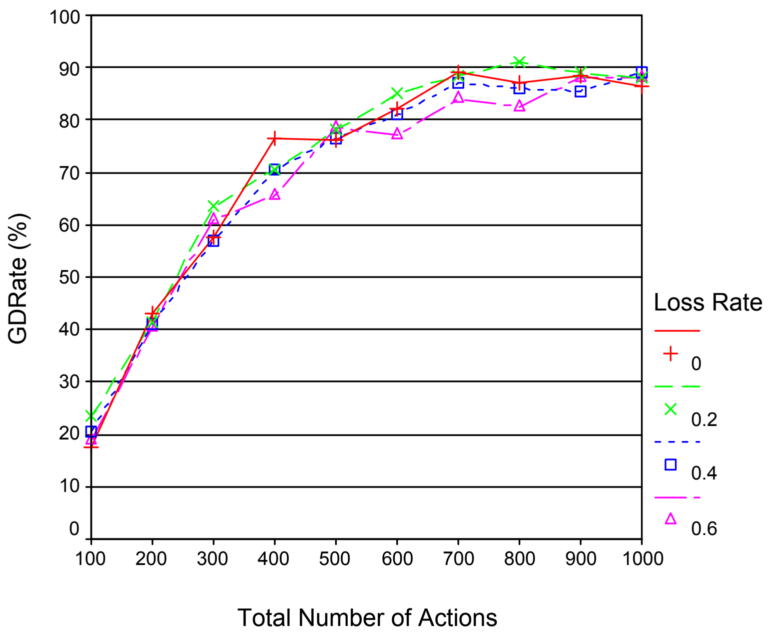


Fig. 8. Total number of actions vs. GDRate

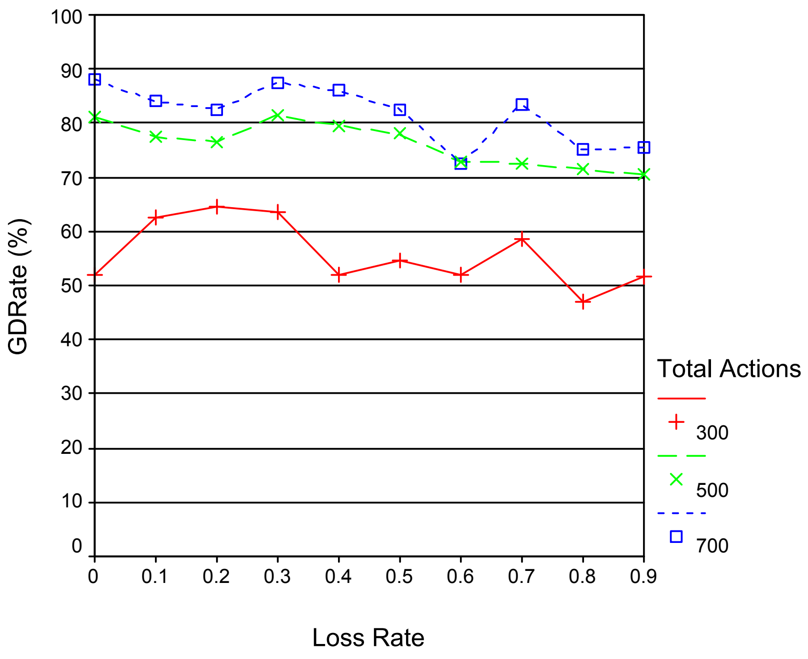


Fig. 9. Loss rate vs. GDRate

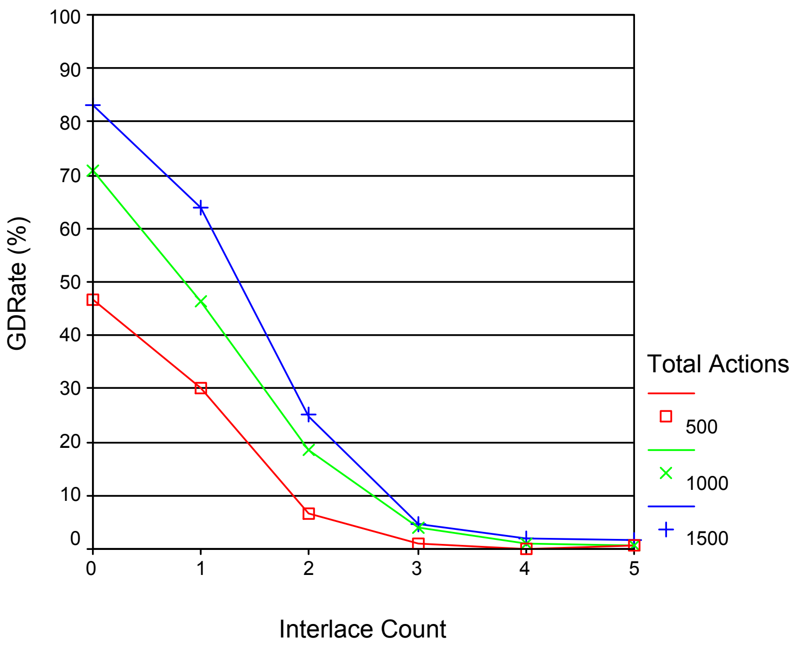


Fig. 10. Interlace count vs. GDRate

The third experimental result is shown in Fig. 10. The x-axis represents the interlace count from zero to five. The y-axis represents the *GDRate*. Three different curves represent the results with the total number of actions of 500, 1000, and 1500. The curves show that the *GDRate* decreases rapidly as the interlace count increases. The setting of the interlace count causes the prior and later actions to be interlaced with actions of other groups in the action log. If the interlace count equals two, four actions of a group are interlaced with other groups. This prevents the discovery of action groups. Therefore, the *GDRate* approaches zero when the interlace count is greater than two. In real life, if the actions of groups are interlaced less frequently, the AGD algorithm can still discover most of the action groups.

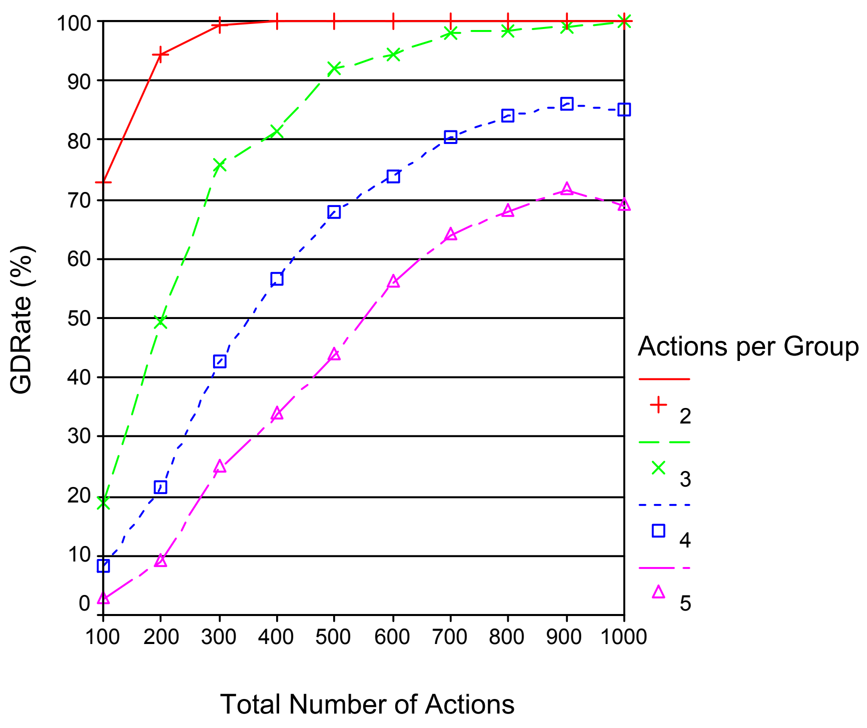


Fig. 11. Actions per group vs. GDRate

The fourth experimental result is shown in Fig. 11. The x-axis represents the total number of actions from 100 to 1,000. The y-axis represents the *GDRate*. Four different curves represent the *GDRate* with actions per group of two, three, four, and five. When the actions per group equal two, three, and four, the *GDRate* can reach up to 80 percent when the total number of actions are approximately 140, 380, and 700, respectively. The curves show hat the more actions per group there are, the more total number of actions is needed for discovering all the action groups correctly. The relationship between the actions per group and the total number of actions can be seen in Table 6.

In Table 6, the data of the first three rows were computed according to the curves in Fig. 11. The last row is a predicted result based on the first three columns. The column of distinct actions was computed by multiplying the actions per group by 20 groups. The approximate total number of actions for the GDRate equaling 80 percent is listed in the third column. The average number of times an action appears in the approximate total number of actions is shown in the fourth column. The ratios of the average numbers 6.3 and 11.6 to 3.5 are listed in the last column. The ratio for five actions per group, according to the increasing trend of the ratios, was calculated using linear approximation. The predicted total actions were 1,733. This shows that when the actions per group are increased by one, the total number of actions must be increased by about 1.7 times to achieve the same GDRate.

Table 6. The relations between actions per group and total actions

Actions per group	Distinct actions	Approximate total number of actions for *GDRate*=80%	Average number of times	Ratio
2	40 (2 actions×20 groups)	140	3.5	1
3	60 (3 actions ×20 groups)	380	6.3	1.8
4	80 (4 actions ×20 groups)	700	11.6	3.3
5	100 (5 actions ×20 groups)	1733 (predicted)	17.33	4.95

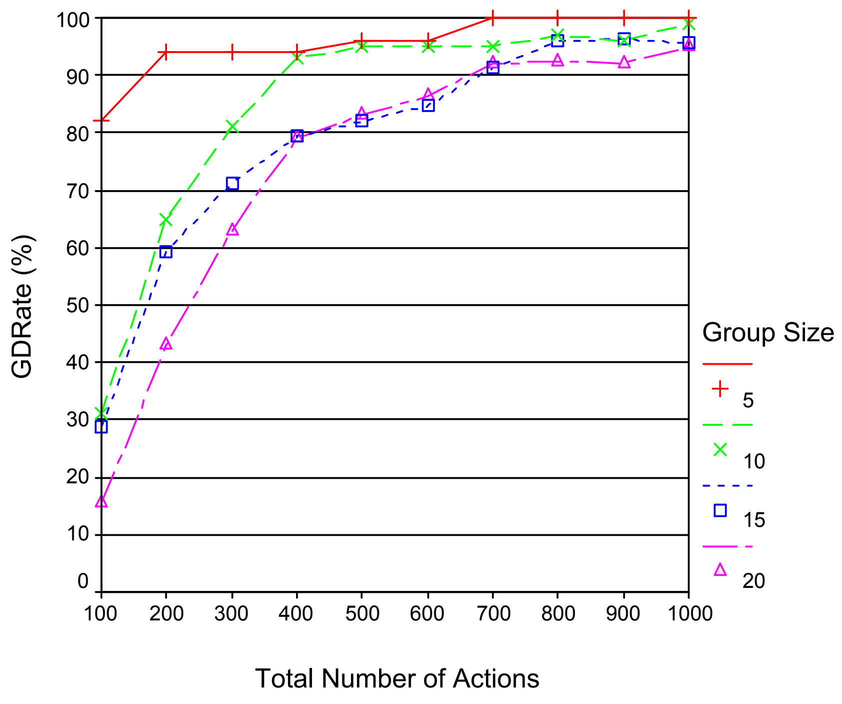


Fig. 12. Group size vs. GDRate

The fifth experimental result is shown in Fig. 12. The x-axis represents the total number of actions from 100 to 1,000. The y-axis represents the *GDRate*. Four different curves represent the *GDRate* with the group sizes 5, 10, 15, and 20. The curves show that the more group sizes there are, the more total number of actions is needed for discovering most of the action groups. The *GDRate* is over 90 percent for all the four different group sizes while the total number of actions is larger than 700. If the action sequence is increased by a rate of 50 actions per day, 14 days are needed to collect 700 actions. This is acceptable for inhabitants to allow the AGD algorithm to discover their action groups.

The efficiency of the AGD algorithm, according to the above experimental results, is summarized as follows:

- The *GDRate* of the AGD algorithm can reach 80 percent for 600 actions, according to the first experimental result.
- The larger the actions per group or group sizes there is, the longer the action sequence is needed to reach a certain *GDRate*.
- The AGD algorithm can still discover action groups without being influenced by the loss rate. On the other hand, the alternation of action groups has a great impact on the *GDRate*. This means that the discovery of action groups from an action sequence with incomplete information is much easier than that with complete but faulty information.

In advance, a realistic evaluation is also obtained here. A real human action sequence was recorded via a mobile device for three months. There were 19 distinct actions and 1,259 actions collected in the sequence. After the process of AGD, three groups (total seven actions) were discovered and verified by the user successfully. However, these 19 actions are distributed in a three-floor building. There are only seven actions average in a floor. It decreases the possible number of action groups to be discovered. Although the number of discovered groups is small, it illustrates that AGD is work on discovering action groups.

5 Conclusion and Future Work

An inhabitant's interactions with devices in a smart home are usually in an arbitrary order. In fact, these actions form a set of action groups. The set of groups should be discovered before the action prediction in order to increase the prediction accuracy. In this paper, AGD algorithm is proposed to discover the action groups from the action sequence. It is mainly based on the positive and reverse 1-order Markov models. The experimental results show that the AGD algorithm can discover most action groups in various situations.

In the future, a group prediction approach based upon the action groups will be designed. Current action prediction techniques focus on the prediction of next action. A part of action sequence is used for training the proposed model. The model is then used on action prediction. The action sequence is not influenced by the action prediction. It is easily to compute the prediction accuracy. Oppositely, the action groups discovered by AGD can be used for the prediction of the next group. Those actions of the next group can be taken automatically. The action sequence may be influenced by the group prediction. For example, the actions from two groups are interleaving in the action sequence. The group prediction may cause the actions of the next group can be taken together. That is, the interleaving of actions may be disappeared and the final action sequence is influenced by the group prediction. Consequently, the computation of prediction accuracy has to be redefined. This is one of the problems on the design of group prediction approach. On the other hand, the incorporation of AGD and current action prediction techniques is also studied in order to improve the prediction accuracy.

References

[1] D. J. Cook, M. Youngblood, and E. O. Heierman, "MavHome: An Agent-Based Smart Home," in *Proceedings of the First IEEE Int'l Conf. on Pervasive Comp. and Comm. (PerCom'03)*, Vol.23, No.26, pp.521-524, March 2003.

[2] E. O. Heierman and D. J. Cook, "Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns," in *Proceedings of the Third IEEE Int'l Conf. on Data Mining (ICDM'03)*, Vol.19, No.22, pp.537-540, Nov.2003.

[3] S. K. Das, D. J. Cook, et al., "The Role of Prediction Algorithm in the MavHome Smart Home Architecture," *IEEE Wireless Communication*, Vol.9, No.6, pp.77-84, 2002.

[4] V. N. Padmanabhan and J. C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *Computer Comm. Rev.*, Vol.30, No.3, pp.22-36, July 1996.

[5] P. Pirolli and J. Pitkow, "Distribution of Surfer's Paths through the World Wide Web: Empirical Characterization," *World Wide Web*, Vol.2, No.1-2, pp.29-45, Jan. 1999.

[6] Y. J. Lin, H. A. Latchman, and M. Lee, "A Power Line Communication Network Infrastructure for the Smart Home," *IEEE Wireless Communication*, Vol.9, No.6, pp.104-111, 2002.

[7] A. Roy, S. K. D. Bhaumik, A. Bhattacharya, et al., "Location Aware Resource Management in Smart Homes," in *Proceedings of the First IEEE Int'l Conf. on Pervasive Computing and Communication (PerCom'03)*, Vol.23, No.26, pp.481-488, March 2003.

[8] R. Kango, P. R. Moore, and J. Pu, "Networked Smart Home Appliances – Enabling Real Ubiquitous Culture," in *Proceedings of IEEE Fifth International Workshop on Networked Appliance*, Vol.30, No.31, pp.76-80, Oct. 2002.

[9] D. Valtchev and I. Frankov, "Service Gateway Architecture for a Smart Home," *IEEE Communications Magazine*, Vol.40, No.4, pp.126-132, April 2002.

[10] H. Jahnke and J. Stier, "Facilitating the Programming of the Smart Home," *IEEE Wireless Communications*, Vol.9, No.6, pp.70-76, Dec. 2002.

[11] V. Stanford, "Using Pervasive Computing to Deliver Elder Care," *IEEE Pervasive Computing*, Vol.1, No.1, pp. 10-13, 2002.

[12] S. Helal, B. Winkler, C. Lee, et al., “Enabling Location-Aware Pervasive Computing Applications for the Elderly,” in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communication (PerCom’03)*, pp.531-536, March 2003.

[13] S. P. Rao and D. J. Cook, “Predicting Inhabitant Action Using Action and Task Models with Application to Smart Homes,” *International Journal on Artificial Intelligence Tools*, Vol.13, No.1, pp.81-99, 2004.

[14] N. Noury, G. Virone, and T. Creuzet, “The Health Integrated Smart Home Information System (HIS^2): Rule Based System for the Localisation of a Human,” in *Proceedings of the Second Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine & Biology*, pp.318-321, May 2002.

[15] A. P. Pons, “Web-application centric object prefetching,” *The Journal of Systems and Software*, Vol.67, No.3, pp. 193-200, 2003.