Chapter 8
Filtering Order Adaptation Based on Attractor Selection for Data Broadcasting System

**Shinya Kitajima, Takahiro Hara,
Tsutomu Terada, and Shojiro Nishio**

Summary Recent spread of different data broadcasting services leads to provide enormous and various heterogeneous data. Since data that a client needs are a part of them, there has been an increasing interest in information filtering techniques where a client automatically chooses and stores the necessary data. Generally, when a client performs filtering, it applies some filters sequentially, and the time required for filtering changes according to the order of filters. On the other hand, in recent years, there have been many studies about attractor selection which is an autonomous parameter control technique based on the knowledge from living organisms. In this chapter, in order to reduce the load for filtering, we propose novel methods which adaptively change the order of filters according to the change in broadcast contents. These methods adaptively decide the control parameters for filtering by using attractor selection.

8.1 Introduction

Recent spread of different data broadcasting services leads to provide enormous and various heterogeneous data. In a broadcast system, while the server can

S. Kitajima (✉) · T. Hara · S. Nishio
Dept. of Multimedia Eng., Grad. School of Information Science and Technology,
Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
e-mail: lastis@infoseek.jp

T. Hara
e-mail: hara@ist.osaka-u.ac.jp

S. Nishio
e-mail: nishio@ist.osaka-u.ac.jp

T. Terada
Dept. of Electrical and Electronics Eng., Grad. School of Science and Technology, Kobe
University, 1-1 Rokkodai, Nada, Kobe 657-8501, Japan
e-mail: tsutomu@eedept.kobe-u.ac.jp

F. Xhafa et al. (eds.), *Complex Intelligent Systems and Their Applications,*
Springer Optimization and Its Applications 41,
DOI 10.1007/978-1-4419-1636-5_8,

broadcast large amount of data at a time, clients are typically mobile terminals whose storage are limited (Belkin and Croft 1992; Bell and Moffat 1996; Sawai et al. 2004). Therefore, there has been an increasing interest in information filtering techniques which automatically choose and store necessary data on the client's storage.

Generally, when a client performs filtering, it applies some filters sequentially. The time required for filtering changes according to the order of filters, since the number of data items that match each filter and the filtering load are different among filters. When the filtering load is high, the filtering speed might become slower than the receiving speed of data. Thus, in an information filtering system, how to determine the order of filters is a crucial problem.

On the other hand, in recent years, there have been several studies about *attractor selection* which is an autonomous parameter control technique based on the knowledge from living organisms (Kashiwagi et al. 2006; Leibnitz et al. 2005, 2009). By using attractor selection, the system can control parameters depending on the situation autonomously, and thus it can cope with changes of the system environment flexibly.

In this chapter, in order to reduce the load for filtering process, we propose novel methods which adaptively change the order of filters adapting to the change in broadcast contents (Kitajima et al. 2009). These methods adaptively decide the control parameters for filtering by using attractor selection. Furthermore, we show the results of simulation experiments, from which we confirm that the proposal methods improve the load for filtering process compared with other methods.

The remainder of this chapter is organized as follows. Section 8.2 describes the outline of an information filtering system, and Sect. 8.3 introduces attractor selection. Section 8.4 explains our proposed methods in details. Section 8.5 evaluates the performance of our methods. Finally, we conclude the chapter in Sect. 8.6.

8.2 Information Filtering System

8.2.1 Mobile Environment

There are several data broadcasting services that have been already available, e.g., those using a surplus band of terrestrial broadcasting, news distributions on the Internet, and bidirectional data services using satellite broadcasting. In such data broadcasting services, the server can send enormous information to a large number of users at a time. However, the data wanted by a user are generally just a small part of the broadcast data.

In this chapter, we assume an urban data broadcasting service in town which is thought to be common in the near future. Figure 8.1 shows a system environment assumed in this chapter. In this environment, mobile users equipped with portable devices such as PDAs and smartphones (mobile clients) walk in town and receive broadcast data via the wireless channel from the nearest server. Some conventional

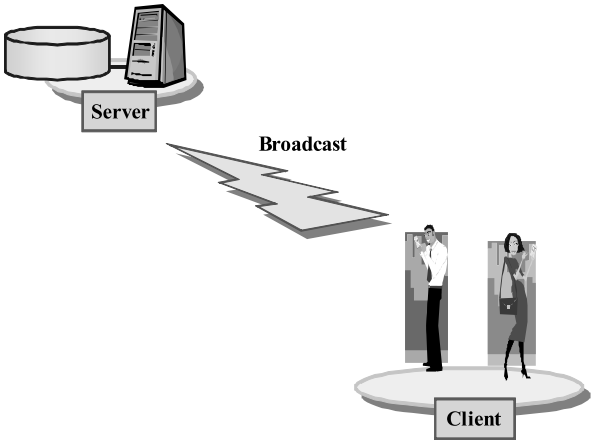

Fig. 8.1 Assumed environment

works such as (Acharya et al. 1995) also assume such an information broadcasting system.

Broadcast contents are mainly text data, and the data of various genres such as real time information like news and weather forecast, local store information, and event information are broadcast. Since mobile clients have a limit for the storage, information filtering to automatically choose the necessary information for users is highly required. We call such a system as an *information filtering system*.

8.2.2 Filtering Architecture

In the information filtering system shown in Fig. 8.2, each client stores broadcast data items once into its receiving buffer, then performs filtering operations when the number of the received items reaches the predetermined constant, and stores only the necessary data items on the storage. Here, we show an example of filtering broadcast data. If a user wants to get data on today's news about sports, the system performs filtering operations by using three kinds of filters to get contents whose (i) category is "news," (ii) issue date is today, and (iii) topic is "sports."

There are various kinds of filters, e.g., a filter which gets items that match specified category or keyword, a filter which gets items that are given a timestamp of the particular period of time, a filter which gets items of high relevance by using the cosine correlation between the user's preference and items (Salton and McGill 1983). The load of applying these filters is different with each other. For instance, the load of calculating the cosine correlation is heavier than that of simple keyword matching.

Moreover, some filters are often applied at the same time as shown in the above example. If the filters do not include ranking operations and do include only se-

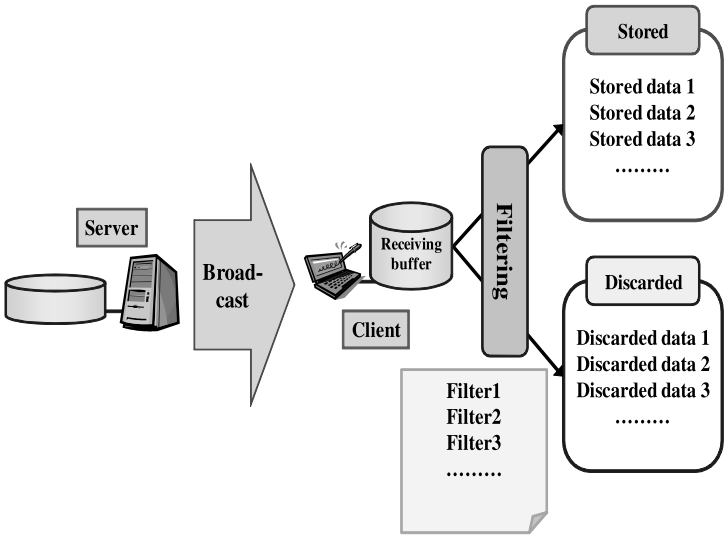

Fig. 8.2 Information filtering system

lection operations, the order of applying filters does not affect the result of filtering (Sawai et al. 2004).

8.2.3 Filtering Cost

If there are multiple filters to apply, the order of applying filters influences the *filtering cost* since the number of data items that match each filter and the *processing cost* of the filter are different among filters. Here, the filtering cost represents the time to perform the filtering operations as a numerical value, and the processing cost means the processing time of the filter per data item. The time to apply a filter is proportional to the processing time of the filter per data item and the number of data items that are applied the filter.

Figure 8.3 shows an example that the filtering cost changes according to the order of filters. In this figure, Tables (a) and (b) show a case in which there are same five broadcast data items stored in the receiving buffer of a client, but the order of applying filters is different. Here, let us assume that two attributes are attached to each data item (Attribute1: category, Attribute2: keyword) that represent the contents of the item. For instance, item 1 belongs to category "news," and its keyword is "sports."

In Table (a), a filter to select data items in category "news" is applied to five items at first, and then another filter to select data items with keyword "sports" is applied to two items that are selected by the first filter. Let us also assume the processing cost of both filters is 1. In this case, the total cost becomes 7. On the other hand, in Table (b), the keyword filter ("sports") is applied to the five items at first, and then

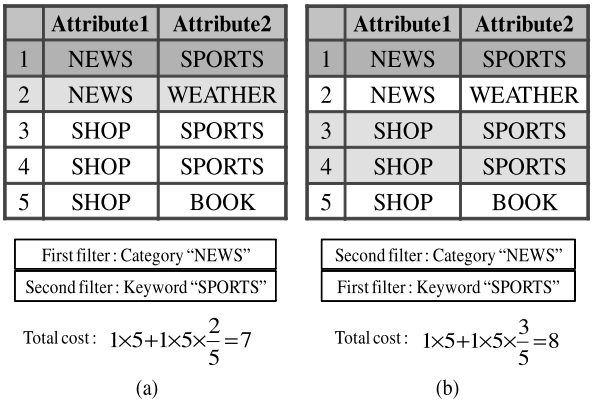

Fig. 8.3 Calculation of filtering cost

the category filter ("news") is applied to three items that are selected by the keyword filter. In this case, the total cost becomes 8. In this way, the filtering cost changes according to the number of data items that match each filter and the processing cost of the filter.

If the processing speed of the filters becomes lower than the receiving speed, the receiving buffer overflows since data are continuously broadcast. In addition, users use their mobile terminals not only to receive the broadcast data but also for other services such as a navigation tool and Video on Demand (VOD). Therefore, the filtering cost should be as small as possible.

In our assumed environment, there are various broadcast data whose contents dynamically change, such as real time information like news, weather, and local store and event information. Moreover, users' demand also dynamically changes. In such an environment, a method that can adaptively and dynamically decide the order of filters is needed.

If the time to apply filters to data items is longer than the time to receive these data items, it is impossible to apply filters to all data items. Therefore, enough processing speed is required to perform filtering. Moreover, enough storage is required since each client stores broadcast data items once into its receiving buffer. The required processing speed changes according to the bandwidth of the broadcast channel and the size of a data item, and the required storage changes according to the size of the receiving buffer and the size of a data item.

8.3 Attractor Selection

8.3.1 Adaptive Response by Attractor Selection

In this subsection, we describe the outline of the attractor selection mechanism, which has been proposed in Kashiwagi et al. (2006). The authors claim that or-

ganisms form a complicated network system having the networks of many hierarchies such as gene, protein, and metabolism, which they call the *organism networks*. When different organism networks meet together, they reach to a stable state (*attractor*) while changing their structure and route and form an organism symbiosis network. This mechanism has many properties such as expansibility, autonomy, toughness, flexibility, adaptability, and variety, which are also needed in an information network and system. Here, "symbiosis" means that multiple different organisms interact with each other and live by supplying the properties to others which they do not have mutually.

It is necessary to adapt to a new environment flexibly while two kinds of organisms without having met before process to form symbiosis relations. However, they have not experienced this environment change in the past, so that they cannot prepare for a hereditary program corresponding to it.

By conventional studies, it becomes clear that transition from an original stable state to a new stable state by the reorganization of the gene metabolism network (three classes of networks of gene, protein, and metabolism), and interaction between the cells by the chemical substance are important. Based on this, the authors suggest a new mechanism called the adaptive response by attractor selection.

In Kashiwagi et al. (2006), to represent a complicated gene metabolism network simply, a model having double feedback loops is defined as follows:

$$\frac{dm_1}{dt} = \frac{syn(act)}{1+m_2^2} - deg(act)\cdot m_1 + \eta_1, \tag{8.1}$$

$$\frac{dm_2}{dt} = \frac{syn(act)}{1+m_1^2} - deg(act)\cdot m_2 + \eta_2, \tag{8.2}$$

$$syn(act) = \frac{6act}{2+act}, \tag{8.3}$$

$$deg(act) = act. \tag{8.4}$$

Here, m_1 and m_2 are mRNA densities made by operons 1 and 2, where an operon is one of the functional units existing on a genome; η_1 and η_2 in the third term on the right side in (8.1) and (8.2) are noises. Equations (8.1)–(8.4) show that when the activity *act* is high, the mRNA density rarely changes. This is because the first term on the right side in (8.1) and (8.2) becomes dominant. On the other hand, when the activity is low, the third term in (8.1) and (8.2), i.e., the noise, becomes dominant, and the system tries to transit another stable state. The activity *act* changes according to the following equation:

$$\frac{dact}{dt} = \frac{pro}{((\frac{Nut\_th_1}{m_1+Nut_1})^{n_1}+1)\times((\frac{Nut\_th_2}{m_2+Nut_2})^{n_2}+1)} - cons\times act. \tag{8.5}$$

Here, Nut_1 and Nut_2 are the supply densities of nourishment from the outside for operons 1 and 2; and Nut_th_1 and Nut_th_2 are their thresholds. *pro* and *cons* are the coefficients of production and consumption of the activity, and n_1 and n_2 are appropriate constant numbers.

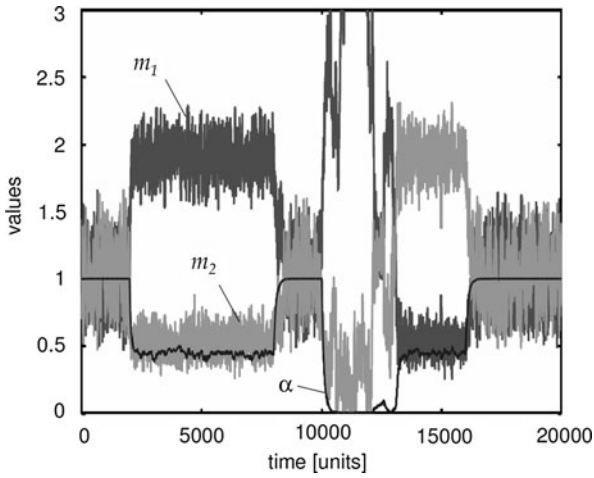

Fig. 8.4 Response of the double feedback loop

Figure 8.4 shows the responses of these double feedback loops. It can be seen that when the outside supply for one nourishment is cut, the attractor making up for the lack is selected. There are two absorption domains in this environment, but only an appropriate attractor is selected. This is because the fluctuation by noises becomes large when the environment becomes worse and activity *act* becomes low, and then, the system is absorbed by that attractor while it approaches to the attractor and recovers the activity. This behavior is an environmental adaptation by attractor selection.

8.3.2 Advantages of Attractor Selection

The fully premeditated construction of systems has become impossible with the rapid large-scaling and complexifying of recent information systems. Therefore, in recent years, system management techniques to adapt for changes of the environment flexibly and autonomously have become crucial.

For instance, in the field of network design, conventional systems have been fully designed to optimize performance and efficiency for specific (predictable) situations. However, such an approach does not work well in recent complicated systems, because it takes long time or sometimes impossible to recover from a large-scale network failure, especially, unknown type of failures. Therefore, to cope with unpredictable changes of the system environment, e.g., system failures and change of system inputs, another design approach is required, in which each component in the system behaves to maintain a stable state and adapts for the environmental change flexibly and autonomously. By this approach, the system can maintain a stable state and offer a service of good quality in a highly dynamic environment, although the performance may not be optimized.

As the system becomes large-scale and more complicated, it becomes impractical to know in advance all events happened in the system and their factors, e.g., reasons and how to deal with them.

Fuzzy reasoning (Lee 1972) accumulates human knowledge as a knowledge database with *If-Then-Else* rules and performs recognition, control, and reasoning by the reasoning engine. While it is based on human experiences, it still requires advanced construction of rules, and thus, the above-mentioned problem is not solved.

Neural networks (Haykin 1989) perform pattern recognition by optimizing parameters in neurons. However, since they are based on machine learning, they cannot quickly adapt to a situation that has never been met and is hard to be predicted.

On the other hand, a genetic algorithm (Goldberg 1992) converts engineering data into the form of gene code and optimizes the system by imitating heredity processes that occur in organisms such as mutation, recombination, and optimal choice. Since it has both two aspects of random search and optimal choice, it is different from approaches that cannot get away from the local minimum such as the steepest descent method (Brooks et al. 1983). However, it basically assumes a well-formulated system, and thus, it cannot handle unknown changes occurred in the system.

Simulated annealing (Kirkpatrick et al. 1983) is an approach that examines multiple neighboring solutions of the current solution randomly and decides probabilistically which neighboring state to transit. It also has a mechanism to prevent from falling into the local minimum. However, since it searches for the optimal solution heuristically, it generally takes much time to converge and cannot cope with frequent changes of the environment.

As mentioned above, conventional approaches cannot fully cope with unknown changes in the system flexibly and quickly.

On the other hand, attractor selection has advantages that it can cope with unknown changes, and its calculation time is much shorter than conventional heuristic approaches. Since the contents of broadcast data are continuously changing in our assumed system environment, attractor selection is suitable for the problem of determining the order of filters which we address in this chapter.

8.4 Proposed Methods

In this section, we propose four methods that can adapt to the change of broadcast contents and reduce the filtering cost by using attractor selection to control the parameters in deciding the order of filters.

8.4.1 Attractor Selection (AS) Method

In the AS method, a client determines *filter selection priority S* by using attractor selection, where *S* consists of a list of filters and defines the order of applying the fil-

ters. Here, let us denote n as the number of applying filters and $S_{i,j}$ as the filter selection priority of applying filter F_i $(i = 1, 2, \ldots, n)$ at jth position $(j = 1, 2, \ldots, n)$ in S.

In the following, we describe the AS method in detail.

Calculation of the Filtering Cost We define the ratio of data items which are discarded by applying filter F_i as *decrease ratio* D_i. D_i is calculated by the following equation using the number of data items, d_i, discarded by F_i, and the number of data items, a_i, that are applied F_i:

$$D_i = \frac{d_i}{a_i}. \tag{8.6}$$

When a client uses the AS method in a real environment, it actually applies the filters to the broadcast data items and uses the elapsed time to perform filtering as the filtering cost. However, in our simulation evaluation, a client cannot apply filters actually since we use pseudo-data. Thus, we generalize the filtering cost and define the processing cost of each filter F_j as c_j. c_j represents the processing time per data item when a client applies F_j.

The total cost C for applying n kinds of filters in a certain order to N data items is calculated by the following equation:

$$C = \sum_{j=1}^{n} \left(c_j N \prod_{k=1}^{j-1} D_k \right). \tag{8.7}$$

Calculation of the Activity In the AS method, the activity α is defined by using C, since the system performance is considered better when the filtering cost is lower. Here, the minimum value of C among the last x results of filtering is denoted by $C_{\min}$. Then, the activity α is calculated by the following equation, where the activity becomes higher when the filtering cost approaches the minimum cost:

$$\frac{d\alpha}{dt} = \delta \left(\left(\frac{C_{\min}}{C} \right)^{\lambda} - \alpha \right). \tag{8.8}$$

Here, δ and λ are scale factors to control the adaptation rate and the value of activity. Note that α ranges $0 \leq \alpha \leq 1$.

Calculation of the Selection Priority The selection priority $S_{i,j}$ is defined by the following equation, which comes from the approaches in Leibnitz et al. (2005):

$$\frac{d}{dt} S_{i,j} = \frac{syn(\alpha)}{1 + S_{\max,j}^2 - S_{i,j}^2} - deg(\alpha) S_{i,j} + \eta_{i,j}, \tag{8.9}$$

$$syn(\alpha) = \alpha \left[\beta \alpha^{\gamma} + \phi^* \right], \tag{8.10}$$

$$deg(\alpha) = \alpha, \tag{8.11}$$

$$\phi(\alpha) = \frac{syn(\alpha)}{deg(\alpha)}, \tag{8.12}$$

$$\phi^* = \frac{1}{\sqrt{2}}. \tag{8.13}$$

Here, $\eta_{i,j}$ is a random number, and β and γ are scale factors (constants). $S_{i,j}$ ranges $0 \leq S_{i,j}$. Note that for $j \geq 2$, $S_{i,j}$ is set as 0 if F_i is already selected, since it is meaningless to apply the same filter more than once.

When the filtering cost is low and the activity is high, the selection priority rarely changes since the first term on the right side in (8.9) becomes dominant. However, when the filtering cost becomes high and the activity becomes low, the third term, i.e., noise, becomes dominant, and the system tries to transit another stable state. That is, the system adapts to the change of the broadcast contents.

Flow Chart Figure 8.5 shows the flow chart of the procedure performed by a client every time when N data items are stored in its receiving buffer in the AS method. We define one cycle of all the steps in this figure as a unit of filtering.

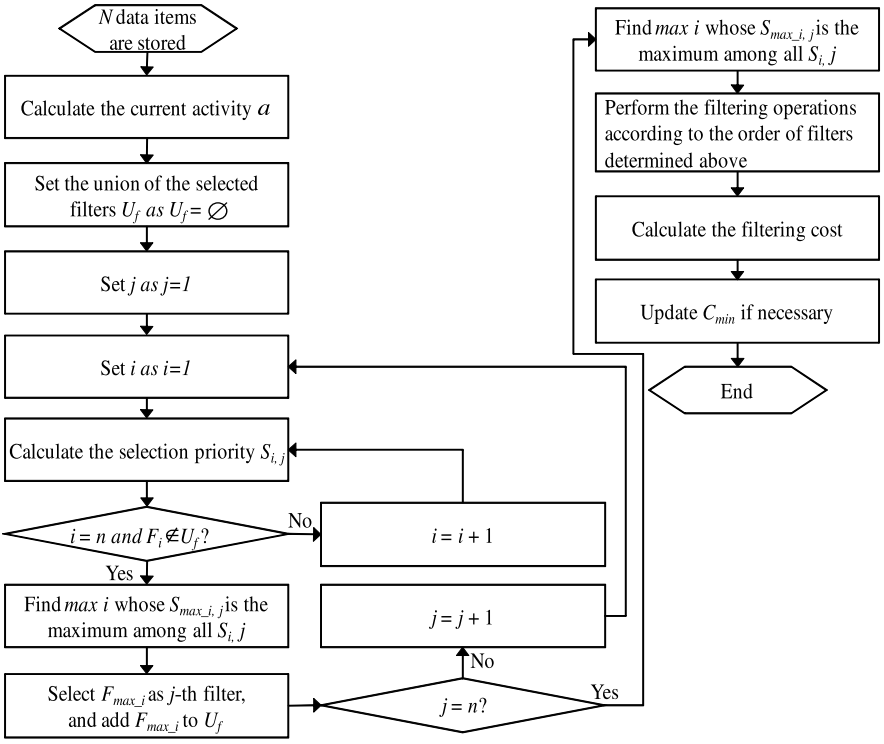

Fig. 8.5 Flow chart of the AS method

Problems of the AS Method Generally, a user's preference changes as time passes. When a user's preference changes, the condition of filtering changes (e.g., "I want to get the information about news, though I am receiving the information about sports"), and the number of filters also might change (e.g., "I want to get the information about baseball, though I am receiving the information about sports").

The AS method can deal with the changes in keywords for filtering and the parameters for the thresholds in filters in the same way as the case that broadcast contents change.

Moreover, the AS method can deal with the changes in the number of filters by adding or reducing the selection priority corresponding to these filters. However, in this case, the upper limit and the lower limit of the filtering cost change largely. Since (8.8) and (8.9) do not assume the changes of the upper and lower limits of the filtering cost, the AS method does not work well, i.e., the filtering cost becomes large. Similarly, the AS method does not work well when the processing costs of filters change. This is because the upper and lower limits of the filtering cost also change.

8.4.2 Extended Methods

In this subsection, we extend the AS method and propose three filtering order adaptation methods that consider the changes in the processing cost and the number of filters.

8.4.2.1 AS-M Method

In the AS method, the filtering cost increases when the processing cost c_i or the number of filters is not constant. This is because the filtering cost changes largely according to the changes in the order of the filters and the characteristics of the broadcast contents, which affect the ratio of $C_{\min}$ to C in (8.8). To solve this problem, we extend the AS method by adjusting λ in (8.8) to keep the value of $(C_{\min}/C)^{\lambda}$ as constant.

Specifically, the AS-M (AS using the Maximum filtering cost) method keeps $(C_{\min}/C)^{\lambda}$ as constant by changing λ according to the following equation:

$$\left(\frac{C_{\min}}{C_{\max}}\right)^{\lambda} = 0.02. \tag{8.14}$$

Here, $C_{\max}$ denotes the maximum value of C in the past. Note that 0.02 is found to be an appropriate value from our preliminary experiments.

In this way, the AS-M method controls the impact of the change of the upper and lower limits of the filtering cost in (8.8). Specifically, when the lower limit of the filtering cost becomes smaller and $C_{\min}/C$ in (8.8) becomes smaller, the impact of the change in the lower limit can be eliminated by changing λ smaller according

to (8.14). On the other hand, when the lower limit of the filtering cost becomes larger and $C_{\min}/C$ in (8.8) becomes larger, the impact of the change in the lower limit can be eliminated by changing λ larger.

As a result, the AS-M method can calculate the activity adequately and deal with the changes in the processing costs of filters and the number of the filters related to the change of the user's preference.

8.4.2.2 AS-P Method

The AS-M method can deal with the change of the upper and lower limits of the filtering cost by changing λ in (8.8). However, from (8.8) and (8.9) it is shown that the change of the upper and lower limits can be also handled by changing not λ but $\eta_{i,j}$.

Therefore, the AS-P (AS with Perturbation) method adjusts the balance of the random term in (8.9). This is based on the approach called *attractor perturbation* (Leibnitz et al. 2009), which is an operation method of the random term.

The random term in the AS-P method, $\eta(t)$, is calculated by the following equation:

$$\frac{d\eta(t)}{dt} = \tau\big[\eta\big(1-\alpha(t)\big) - \eta(t)\big]. \tag{8.15}$$

Here, τ denotes the parameter to define the balance of the random term.

The AS-P method controls the random term instead of using the constant noise term $\eta_{i,j}$ in (8.9). According to (8.15), the impact of the random term becomes small when the activity is high, and it becomes large when the activity is low.

As a result, the AS-P method can make the activity more stable when the activity is high. Also, it can transit to another stable state more easily when the activity is low, compared with the AS method.

8.4.2.3 AS-MP Method

Since the AS-M and the AS-P methods can work independently with each other, we suppose that we can further reduce the filtering cost by applying both methods together. Therefore, the AS-MP method combines the AS-M and AS-P methods, where λ and the random term are respectively adjusted according to the AS-M and AS-P methods.

8.5 Evaluation

This section evaluates the proposed methods using simulation studies. The evaluation criterion is the *average filtering cost*, which is the average of the filtering costs for all filtering processes performed during the simulation time.

8.5.1 Simulation Environment

Table 8.1 shows the parameters used in the simulations. In the simulations, the broadcast data and the filtering model are assumed as an information service for mobile clients as described in Sect. 8.2. Moreover, we did not use real broadcast data but use pseudo data to represent various situations by changing parameters. To apply multiple different filters, we attach the same number of attributes as filters to the pseudo-data and the attribute values (e.g., keywords) are used for filtering.

For simplicity, all filters perform selection operations, i.e., only data items that contain the attribute values specified by the client are stored, and other items are discarded. Thus, the filtering results do not depend on the order of applying filters (Sawai et al. 2004). The results of our simulations correspond to the results in other settings where filters whose applying order does not affect the filtering results are assumed. We can also easily consider cases where filters whose applying order does affect the filtering result by introducing a mechanism to take dependencies of the applying order into account when determining the order of filters.

The distribution of attribute values attached to the pseudo-data is determined according to the Zipf distribution. Here, many conventional studies on broadcast information systems also assume the Zipf distribution for distribution of data values (Acharya et al. 1995; Aksoy and Franklin 1998). The Zipf distribution is shown in the following equation:

$$f(r) = \frac{\frac{1}{r}}{\sum_{m=1}^{N_a} \frac{1}{m}}. \tag{8.16}$$

Table 8.1 Parameters

Parameter	Value
Number of times of filtering	50000
Number of filters	5
Number of tags	5
Number of keywords	5
Size of the receiving buffer	5000
Calculation cycle in the optimal method	10000
Calculation cycle in the genetic algorithm	7000
β	0.4
γ	5.0
δ	3.0
η	−0.1–0.1
λ	10
Number of steps in the Runge–Kutta method	10
Bandwidth of the broadcast channel [Mbps]	10
Size of a data item [kByte]	1

Here, N_a denotes the number of attribute values, and r represents the rank when the attribute values are ordered sequentially by the number of data items having that attribute value. $f(r)$ represents the probability that the attribute value of rank r appears. When rank r is low, $f(r)$ becomes high, and the number of data items having that attribute value becomes high. In this chapter, we assume that the attribute values consist of not only the keywords and the categories but also the numerical values such as cosine correlation. When the attribute values are numerical values, it is assumed that the ratio of the number of data items that belong to each interval among the uniformly divided intervals within the total range of the attribute values follows the Zipf distribution.

In the simulations, we change r for each attribute value randomly as time passes. This represents the change of the broadcast contents. We call the cycle of changing r for each attribute value as the *attribute rank changing cycle* and the timing of changing r as the *attribute rank changing timing*. We change the attribute rank changing cycle from 600 to 1400. Moreover, we basically set the processing cost c_i of filter F_i $(i = 1, 2, \ldots, 5)$ as $c_i = 0.6$ $(i = 1, 2, \ldots, 5)$ [ms/data].

It is assumed that the user specifies an attribute value that represents the user's preference to each attribute. The data items whose attribute values match the user's preference are stored, and others are discarded. One filter selects data items whose attribute value matches the user's one attribute. We also assume that the number of attributes attached to each data item equals the number of filters.

In the simulations, we set β, γ, δ, η, and λ as values determined by some preliminary experiments. In the AS-P method, η is set to $-0.5 < \eta < 0.5$. Furthermore, we set $C_{\min}$ as the minimum cost of the past 50 times of filtering.

8.5.2 Comparison Methods

In our simulations, we compared our proposed methods with the following four methods.

Minimum Cost Method: In the minimum cost method, a client calculates the filtering cost for each of all possible $n!$ kinds of orders of filters for every filtering process and adopts the order that gives the minimum cost among them. Note that this method is unrealistic because the computation load is too high to apply it in a real environment. Therefore, we show the performance of this method as a lower bound.

Cyclic Adaptation Method: At a certain calculation cycle, a client once performs filtering using each of all possible $n!$ kinds of orders of filters at the cycle and adopts the order that gives the minimum cost among them. Then, the client adopts the same order of filters until the next calculation cycle. In this method, the filtering cost at the calculation cycle becomes equal to that of the minimum cost method but cannot adapt to the change of broadcast contents until the next cycle.

Note that the load at the calculation cycle is high, since the client has to calculate the costs of all $n!$ kinds of orders. We define the value obtained by dividing the

filtering cost to perform this method once by the calculation cycle as the *optimal order calculation cost*. Moreover, we call the sum of the average filtering cost and the optimal order calculation cost as the *total cost*.

Genetic Algorithm: In the genetic algorithm, similar to the cyclic adaptation method, a client periodically searches an appropriate order of filters according to the genetic algorithm so that the filtering cost becomes less. Then, the client adopts the same order of filters until the next calculation cycle. Specifically, a client selects the order of filters that provides the minimum cost after performing filtering using each of several kinds (much less than $n!$) of orders among $n!$ kinds of orders, while all $n!$ kinds of orders are examined in the cyclic adaptation method. The calculation cost of the genetic algorithm is lower than that of the cyclic adaptation method. However, it cannot always find the order with the minimum cost.

We set the crossover rate as 0.8, the mutation rate as 0.03, the number of children as 10, and the maximum number of generation as 6. Moreover, we use the combination of elitist selection and roulette selection as the selection method, uniform the crossover as crossover method, and inversion mutation as the mutation method.

Random Method: In the random method, a client decides the order of filters at random for every filtering process.

8.5.3 Evaluation Criteria

We use the following three costs as criteria for the evaluation. The unit of all the criteria is millisecond.

Filtering Cost: The time to perform the filtering operations. In our simulation evaluation, a client cannot apply filters actually since we use pseudo-data. Thus, we generalize the filtering cost as (8.7) based on some preliminary experiments. We use this criterion only in Sect. 8.5.4.2.

Average Filtering Cost: The average of filtering costs of all filtering processes performed during the simulation experiments. In other words, it represents the average cost per filtering process.

Total Filtering Cost: The total sum of the average filtering cost and the average calculation cost. The average calculation cost is defined as the average of costs for calculating the order of filters at the calculation cycle in each method. This is based on the actual calculation times in some preliminary experiments. Here, in the AS method and the extended methods, the calculation cost (i.e., the cost for calculating variables such as the activity and the selection priority) is ignored since it is much smaller than the average filtering cost.

8.5.4 Simulation Results

8.5.4.1 Impact of Calculation Cycle in the Cyclic Adaptation Method

Figure 8.6 shows the average filtering cost, the optimal order calculation cost, and the total cost of the cyclic adaptation method when the calculation cycle changes from 1000 to 12000.

The result shows that the average filtering cost when the calculation cycle is 1000 is the lowest. This is because the calculation cycle is short and, thus, the server can adapt to the change of broadcast contents rapidly. The average filtering cost basically becomes higher as the calculation cycle gets longer.

On the other hand, the optimal order calculation cost becomes lower as the calculation cycle gets higher. This is because the longer the calculation cycle is, the less the number of times of calculating the costs is.

The total filtering cost, the sum of the average filtering cost and the optimal order calculation cost, is the lowest when the calculation cycle is 10000. This shows that the average filtering cost and the optimal order calculation cost have a trade-off relation, and the system performance is balanced when the calculation cycle is 10000 in this simulation environment. Thus, we chose 10000 as the calculation cycle in the cyclic adaptation method in the following experiments.

8.5.4.2 Comparison among Methods

Figure 8.7 shows the total filtering cost of each method. From this result, the total filtering costs of the four proposed methods are lower than that of the cyclic adaptation method, the genetic algorithm, and the random method. The average filtering cost of the cyclic adaptation method is slightly lower than the random method. However,

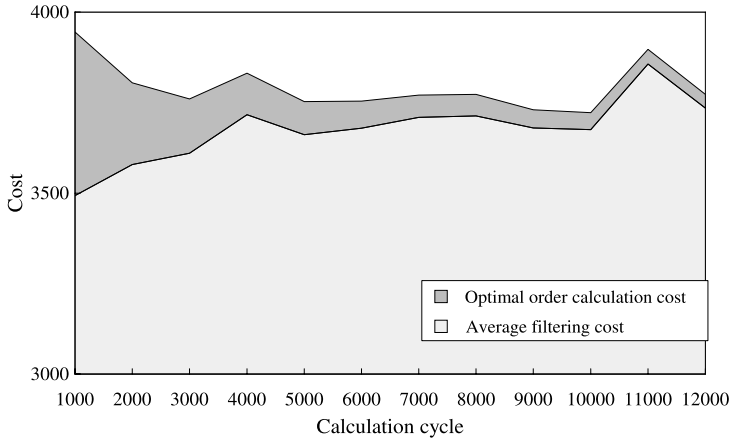

Fig. 8.6 Impact of calculation cycle in the cyclic adaptation method

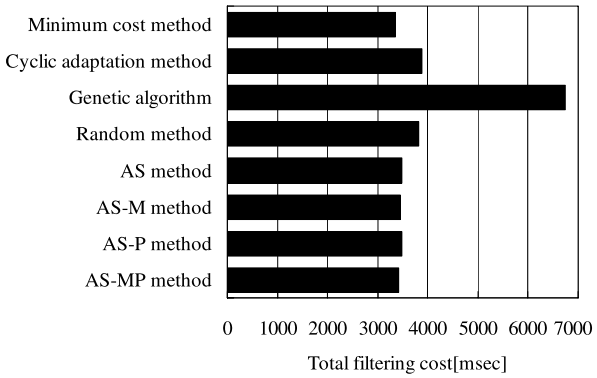

Fig. 8.7 Comparison between methods

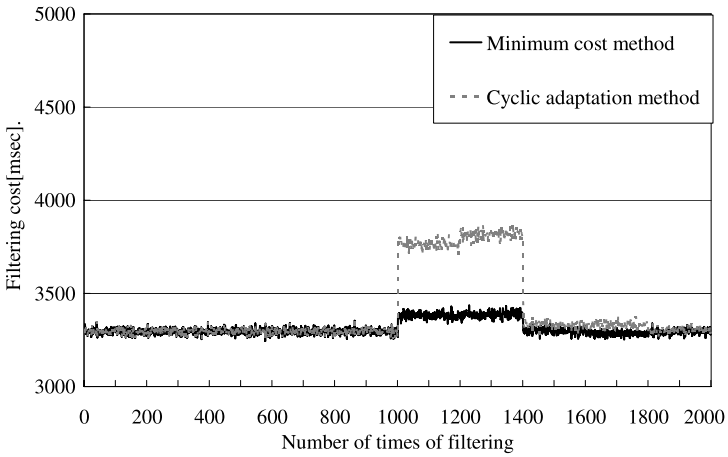

Fig. 8.8 Transition of the filtering cost of the minimum cost method and the cyclic adaptation method

the cyclic adaptation method requires an extra cost to calculate the order of filters, i.e., the optimal order calculation cost, and thus, the total filtering cost is higher than the cost of the random method.

The total filtering cost of the genetic algorithm is much higher than that of the cyclic adaptation method. In the genetic algorithm, the optimal order calculation cost at the calculation cycle is lower than that in the cyclic adaptation method. However, the average filtering cost of filters whose order is determined in the calculation cycle is often far from the minimum. Thus, the total filtering cost of the genetic algorithm becomes higher than that of the cyclic adaptation method.

Figure 8.8 shows the transition of the filtering costs of the minimum cost method and the cyclic adaptation method, and Fig. 8.9 shows the transition of the filtering costs of the AS method and the genetic algorithm. Figures 8.10, 8.11, and 8.12

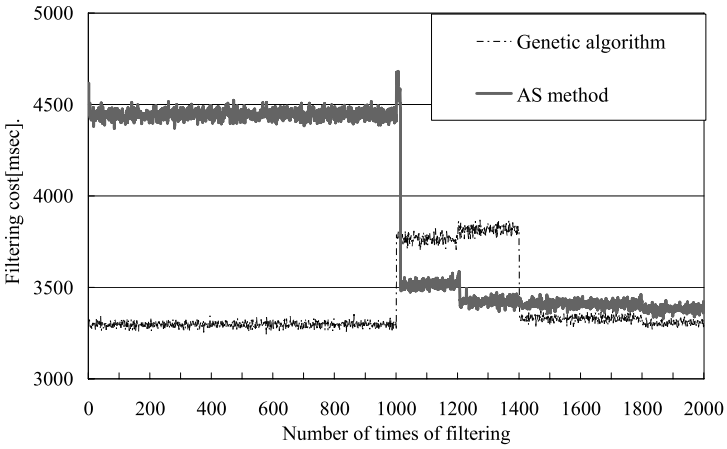

Fig. 8.9 Transition of the filtering cost of the genetic algorithm and the AS method

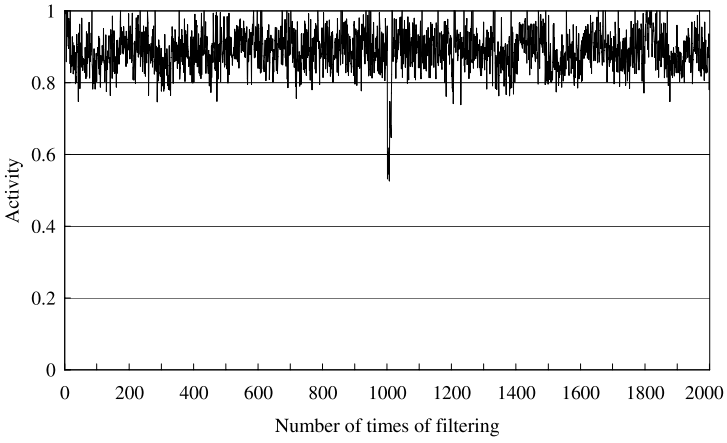

Fig. 8.10 Transition of the activity in the AS method

show the transitions of the activity and the selection priority $S_{i,1}$ $(i = 1, 2, \ldots, 5)$ in the AS method. Due to the limitation of space, we only show the results from the simulation starting time to the time until 2000 times of filtering processes are performed.

From Figs. 8.8 and 8.9, it is shown that the filtering cost of every method except for the minimum cost method changes largely after the time when 1000th filtering process is performed at which broadcast contents change. However, in the AS method, the filtering cost becomes low soon, which shows that our method can adapt to the change of broadcast contents.

Figure 8.10 shows that the activity becomes very low when the broadcast contents change and the filtering cost becomes high. Moreover, Figs. 8.11 and 8.12 show

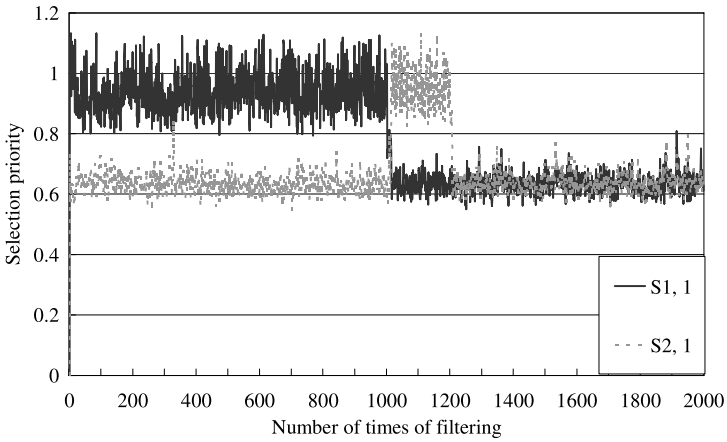

Fig. 8.11 Transition of the selection priority in the AS method ($S_{1,1}$, $S_{2,1}$)

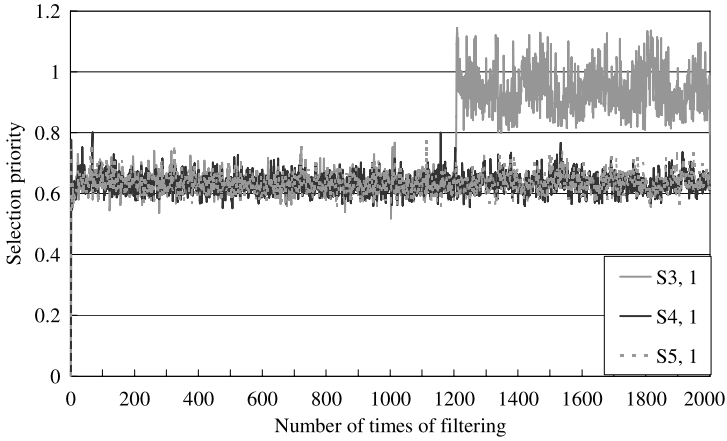

Fig. 8.12 Transition of the selection priority in the AS method ($S_{3,1}$, $S_{4,1}$, $S_{5,1}$)

that when the activity becomes low, the random term in (8.9) influences largely, and thus, the selection priority goes up and down greatly. After a short time, the system transits into a stable state, and the selection priority of a specific filter rises.

In summary, in the AS method, the client changes the order of filters adaptively by using attractor selection to control the selection priority when the broadcast contents change and the filtering cost becomes high. It confirms us the effectiveness of using attractor selection to adapt to the change of the broadcast contents. Here, in the AS method, the activity sometimes becomes low, and the order of filters is changed even when broadcast contents do not change. This is due to the influence of the random term in (8.9).

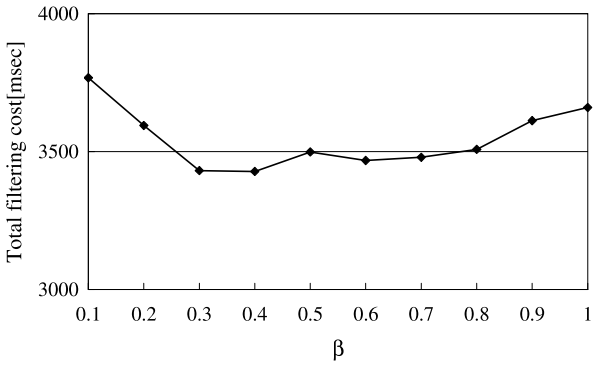

Fig. 8.13 Impact of β

In this simulation setting (as expected), the three extended methods (AS-M, AS-P, and AS-MP) show almost the same performance as the AS method.

8.5.4.3 Impact of β on the AS Method

Figure 8.13 shows the total filtering cost of the AS method when β changes from 0.1 to 1.0. From this result, the total filtering cost is low when $\beta = 0.3$ to 0.8. Here, β is a constant that coordinates the influence of the random term in (8.9). When β is very high, the random term influences little, so that the selection priority and the order of filters do not change even when the filtering cost is high and the activity is low. On the other hand, when β is very low, the random term influences largely, and the AS method acts similarly to the random method.

8.5.4.4 Impact of γ on the AS Method

Figure 8.14 shows the total filtering cost of the AS method when γ changes from 1 to 10. From this result, the total filtering cost is low when $\gamma = 4$ to 7. Here, γ is a constant that coordinates the influence of the activity in (8.9). The influence of the activity becomes small when γ is high. However, in our simulations, the total filtering cost is not much influenced by γ.

8.5.4.5 Impact of δ on the AS Method

Figure 8.15 shows the total filtering cost of the AS method when δ changes from 1 to 7.

The result confirms that the total filtering cost hardly changes even if δ changes, i.e., the impact of δ is very small.

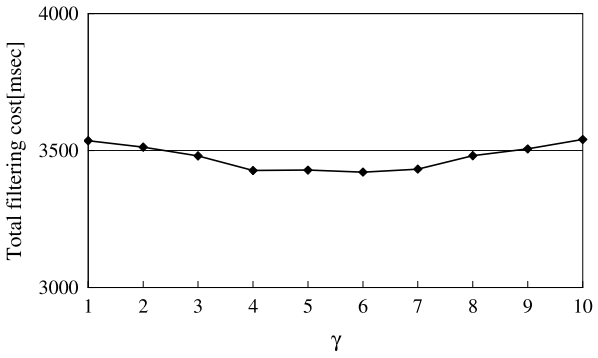

Fig. 8.14 Impact of γ

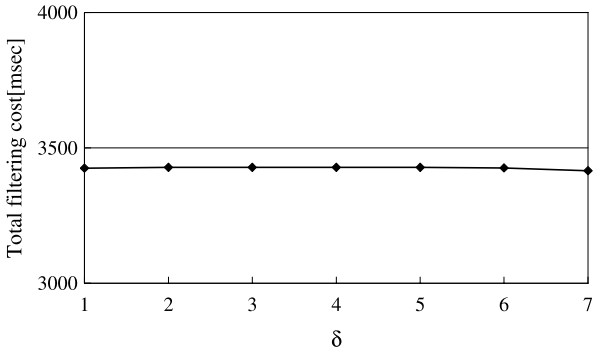

Fig. 8.15 Impact of δ

8.5.4.6 Impact of x on the AS Method

Figure 8.16 shows the total filtering cost of the AS method when x changes from 10 to 100. Here, x is the window size for calculating $C_{\min}$, i.e., the AS method determines $C_{\min}$ as the minimum filtering cost among the last x filtering processes.

From this result, the total filtering cost is low when $x = 30$ to 100. If x is very small, $C_{\min}$ is updated frequently, and the activity tends to be unstable according to (8.8). On the other hand, if x is large, $C_{\min}$ is rarely updated even when the broadcast contents change, and thus, the activity also tends to be unstable.

8.5.4.7 Impact of c_i

Figure 8.17 shows the total filtering cost of each method when changing the processing cost c_i randomly between 0.1 to 2.0 [ms/data] at every 1000th filtering. This figure shows that the AS-M and the AS-P methods reduce the total filtering cost compared with the AS method, the minimum cost method, the cyclic adaptation method, the genetic algorithm, and the random method. Moreover, the AS-MP

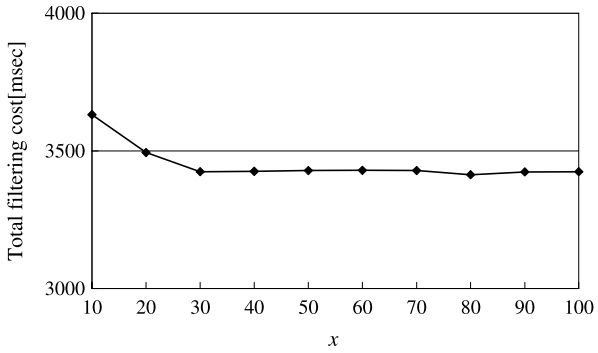

Fig. 8.16 Impact of x

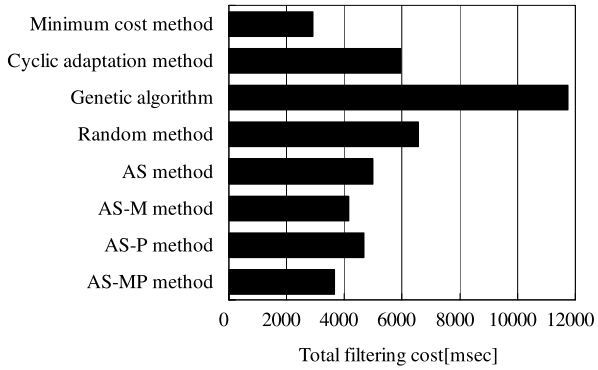

Fig. 8.17 Impact of c_i

method which combines the AS-M and the AS-P methods gives the lowest total filtering cost.

The AS method does not take into account the change of the upper and lower limits, and thus, the order of filters sometimes changes even if the filtering cost is almost same as the lower limit. This is because, according to (8.8), the influence on the activity becomes large when λ is not appropriately chosen.

The AS-M method can calculate the activity appropriately by controlling λ based on the max value of the past filtering costs, even if the processing cost changes. Thus, the AS-M method can adapt to the change in the processing cost.

The AS-P method reduces the total filtering cost, since it can adapt to the change of the filtering cost flexibly by controlling the random term in (8.9). However, the effect of the AS-P method is lower than the AS-M method, since the AS-P method only adjusts the random term in (8.9), while the AS-M method feeds back the change of the upper and lower limits to the activity.

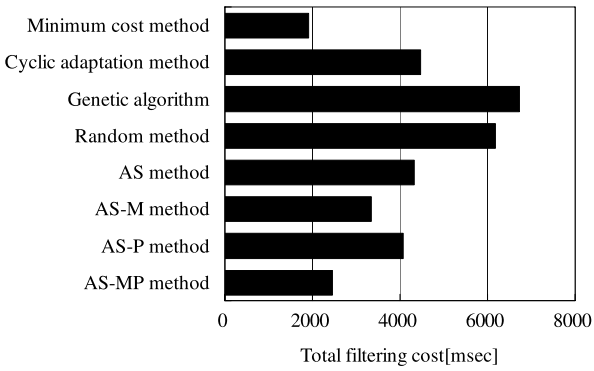

Fig. 8.18 Impact of the number of filters

8.5.4.8 Impact of the Number of Filters

Figure 8.18 shows the total filtering cost of each method when changing the number of filters randomly between 2 to 5 at every 1000th filtering. In this simulation, we set the processing cost c_i as $c_i = [0.2, 0.6, 1.0, 1.4, 1.8]$. Moreover, the number of attributes is set as the number of filters.

From this result, it is also shown that the AS-M and AS-P methods reduce the total filtering cost compared with other methods. Moreover, the AS-MP method gives the lowest total filtering cost. This result confirms us that the extended methods can reduce the total filtering cost by controlling λ and the random term dynamically even when the number of filters changes.

8.6 Conclusions

In this chapter, we proposed a novel method called the AS method that uses attractor selection to control parameters to determine the order of applying filters. With the proposed method, the client adaptively changes the order of filters following the change of broadcast contents to reduce the filtering load. Moreover, we extend the proposed method and propose three methods that take into account the change of the user's preference. The simulation results confirmed us that the proposed methods reduce the filtering cost compared with other methods except for the minimum cost method (lower bound). Furthermore, the extended methods further reduce the filtering cost compared with the AS method.

As part of our future work, we plan to examine the influence of the change of broadcast contents and user's preference on the performance of our methods in more detail.

Acknowledgements This research was partially supported by "Global COE (Centers of Excellence) Program" and Special Coordination Funds for Promoting Science and Technology "Formation of Innovation Center for Fusion of Advanced Technologies: Yuragi Project" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Acharya, S., Alonso, R., Franklin, M., and Zdonik, S.: Broadcast disks: Data management for asymmetric communication environments, in *Proceedings of ACM SIGMOD 1995*, pp. 199–210, May 1995.

Aksoy, D. and Franklin, M.: Scheduling for large-scale on-demand data broadcasting, in *Proceedings of IEEE The Conference on Computer Communications (INFOCOM 1998)*, pp. 651–659, Mar. 1998.

Belkin, N. J. and Croft, W. B.: Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, Vol. 35, No. 12, pp. 29–38, 1992.

Bell, T. A. H. and Moffat, A.: The design of a high performance information filtering system, in *Proceedings of SIGIR 1996*, pp. 12–20, Aug. 1996.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., and Karplus, M.: CHARMM: A program for macromolecular energy, minimization, and dynamics calculations, *Journal of Computational Chemistry*, Vol. 4, pp. 187–217, 1983.

Goldberg, D. E.: Genetic algorithms in search, optimization and machine learning, *Communications of the ACM*, Vol. 35, No. 12, pp. 29–38, 1992.

Haykin, S.: Neural Networks: A Comprehensive Foundation, Addison-Wesley, Reading, 1989.

Kashiwagi, A., Urabe, I., Kaneko, K., and Yomo, T.: Adaptive response of a gene network to environmental changes by fitness-induced attractor selection, *PLoS ONE*, Vol. 1, No. 1, e49, Dec. 2006.

Kitajima, S., Hara, T., Terada, T., and Nishio, S.: Filtering order adaptation based on attractor selection for data broadcasting system, in *Proceedings of International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2009)*, pp. 319–326, Mar. 2009.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P.: Optimization by simulated annealing, *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.

Lee, R. C. T.: Fuzzy logic and the resolution principle, *Journal of the ACM*, Vol. 19, No. 1, pp. 109–119, 1972.

Leibnitz, K., Wakamiya, N., and Murata, M.: Biologically inspired adaptive multi-path routing in overlay networks, in *Proceedings of IFIP/IEEE International Workshop on Self-Managed Systems & Services (SelfMan 2005)*, (CD-ROM), May 2005.

Leibnitz, K., Furusawa, C., Murata, M.: On attractor perturbation through system-inherent fluctuations and its response, in *Proceedings of International Symposium on Nonlinear Theory and its Applications (NOLTA 2009)*, (CD-ROM), Oct. 2009.

Salton, G. and McGill, M. J.: Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983.

Sawai, R., Tsukamoto, M., Terada, T., and Nishio, S.: Composition order of filtering functions for information filtering, in *Proceedings of International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2004)*, pp. 166–171, Jan. 2004.