Constructing a Personal Web Map with Anytime-Control of Web Robots

Seiji Yamada Norikatsu Nagino

CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatsuta, Midori, Yokohama 226-8502, JAPAN
{yamada, nagino}@ymd.dis.titech.ac.jp

Abstract

In this paper, we propose a PWM (Personal Web Map) which is a personal and small database of interesting Web pages to a user, and develop a method to construct it under the user's control of multiple Web robots. Though general search engine with large databases like YaHoo, AltaVista, MetaCrawler are valid, it is important that a user constructs a small, personal database of relevant Web pages to his/her interest like Bookmarks. For such a Web page database, we propose a PWM: a personal database of interesting Web pages to a user which he/she can control its construction. First a user gives keywords indicating his/her interest to a system, and it constructs a PWM concerned with the keywords. For building a useful PWM, it is necessary that a user can interrupt the construction of a PWM anytime and instruct a sub-field in which a PWM should be expanded more. For this function, we develop an anytime-control algorithm for multiple Web robots. A density distribution blackboard is used, and an uniform distributed PWM is built. Whenever a system is interrupted by a user, it provides a valid PWM in terms of keeping search space wide, and indicates many alternatives on which he/she wants more information. From Web pages in a database, document vectors are generated and used to construct a 2D-map of a PWM by using self-organization maps. A user easily recognizes a PWM through the 2D-map, and gives instruction by clicking a node about which he/she wants more detail information. We made experiments by users and found out that our method outperformed breadth-first search for constructing a useful PWM. As results, a PWM system is considered as a promising approach to assist a user in gathering relevant information in the WWW.

1 Introduction

The accessible information through the Internet is increasing explosively as the WWW becomes widespread. In this situation, the WWW is very useful for a user who wants to gather interesting information. However there is a significant issue that a user does not know where the information exists. A practical and simple solution of the problem is to use a search engine like MetaCrawler, AltaVista, YaHoo with the interesting information as a query. The search engine provides a list of relevant Web pages to a user. Unfortunately, since a database of a search engine is very huge and adequate filtering is hard, many Web pages including irrelevant ones may be indicated.

While search engines having large databases are valid, it is important that a user constructs a small, personal database of relevant Web pages to his/her interest like Bookmarks. Since such a Web page database includes only interesting Web pages, irrelevant Web pages are not indicated for a user's query like large search engines. For the personal database of Web pages, it is important that a user can control the construction of a database to customize it. We call such a database *PWM: Personal Web Map*. PWM is a structured database as a classified Web pages and constructed under user's control. A user can determine keywords as input to a PWM system and a PWM is built by gathering relevant Web pages to them and classifying the gathered Web pages. Also a PWM system show a 2D-map of PWM through a Web browser and a user can give feedback on his/her preference to the system. The construction of PWM is interactive since a user can control the granularity on a PWM.

In this paper, we propose a PWM and develop a method to construct it. First we define a PWM as a structured database. Next we explain anytime-control of multiple Web robots which is suitable to build a PWM and make experiments for evaluation. A PWM system is considered as a multi-agent system using a blackboard model. The blackboard indicates the density distribution in every dimensions under vector representation of Web pages. Clustering a PWM and indicating its 2D-map are done by SOM. Finally we make experiments to evaluate a PWM using four Web robots and verify the utility, and limitation is discussed.

WebWatcher[7] and Letizia[9] are able to indicate the

Web pages which a user wants to see next. Using browsing history, they learn to predict useful Web pages for a user. These systems are consider as customizing systems that learn user's preference in searching Web pages. Unfortunately the systems do not build personal Web page databases.

SPHINX[12] is a framework in which a user can achieve the personal crawling tasks. However the customization is on searching of Web pages, not building a personal Web page database. Thus the purpose is different from ours.

Fish search[2] is a distributed search algorithm for gathering relevance information. Agents have energy, which is gained from relevant Web pages and lost from irrelevant pages. Agents having hight-energy can reproduce themselves and others having low-energy may die. ARACHNID[10] is a more excellent system that can gather information by learning information agents. In the similar way to fish search, agents reproduce themselves or die depending on their energy, and obtain energy from relevant Web pages. Furthermore ARACHNID is adaptive to the change of an environment. These systems do distributed control of agents, however the control method is inspired by an artificial life approach and different from anytime-control.

Kohonen's SOM (Self-Organizing Map) have been used to classify documents in the WWW. WEBSOM[5][6] is able to classify articles in network news groups and display the clustered results. WEBSOM just does clustering documents when they are given, and has no contribution to information gathering. In this research, we use SOM for clustering gathered Web pages.

2 Personal Web Map

First we define a PWM (Personal Web Map). PWM is a layered sets of Web pages which is built under user's control. Fig.1 shows a structure of PWM consisting of sets of gathered Web pages called *WPS*s (Web Page Sets). The first layer includes a WPS_{00} including directly relevant Web pages to keywords given by a user as input. WPS_{ij} means the jth *WPS* in the ith-layer, and a layer may include multiple *WPS*s. After gathering, the Web pages are classified by SOM and clusters (white circles in Fig.1) are generated. A user allows a PWM system to construct layers recursively. When he/she points a cluster in a WPS_nj about which he/she wants to know more, a system generates the more detail $WPS_{(n+1)k}$ for the cluster k in a $(n+1)$th layer. With such a procedure, a user is able to control construction of PWM for including detail information about which he/she wants to know more.

In a PWM, each *WPS* contains about hundreds of Web pages. You may claim a PWM must be included by a database of a large search engine. However we often face

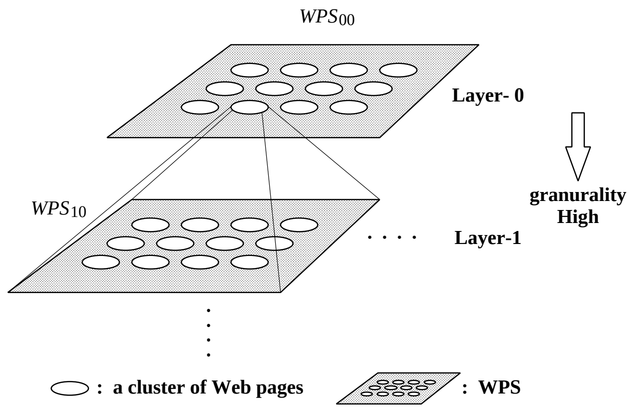

Figure 1. PWM**: Personal Web Map.**

the facts that many interesting Web pages are not captured by such a large search engine. We consider a PWM which is controlled by a user is promising for fast personal search engine with a necessary and sufficient Web page database.

3 Overview of a PWM system

Fig.2 shows the overview of a PWM system. It consists of PWM, a Web page database, a density distribution blackboard, multiple Web robots and an information retrieval system like a search engine. The system provides two different modes in a user's purpose: *Building mode* and *Utilizing mode*. In a building mode, a user can control to construct a PWM. The input of the system are *keywords* on which a user wants to build a PWM, and a system outputs a 2D-map of a PWM to him/her. A user can click a node on the 2D-map for gathering information about it. In a utilizing mode, a user can use the Web page database to search for necessary information. He/she may browse the database or input queries to a search engine. The 2D-map of PWM is also indicated to a user as a structured map like YaHoo's interface.

The two modes do not need to be phased. A user can interrupt the system anytime, and input keywords for a building mode or input queries to use the search engine for a utilizing mode.

Multiple Web robots gather relevant Web pages to keywords. They monitor a density distribution blackboard, and try to gather pages in the most sparse area. This is called *anytime-control*, and we will mention its detail in the next section. The anytime-control makes a database of Web pages uniform and provides a valid 2D-map with many alternatives a use can select. Web pages gathered by Web robots are stored in a Web page database. All Web robots work asynchronously.

A density distribution blackboard is updated with keyword vectors detected from Web pages in the database. A 2D-map is generated depending on document vectors by Self-Organizing Maps.

4 Generating a database for PWM with multi Web robots

How can we construct a PWM? For efficiency, we should use multiple Web robots for building a PWM. Next how should we control them? We propose *anytime-control*: a novel control method for Web robots to effectively gather information for PWM.

4.1 Keyword vectors and a Web page database

In a density distribution blackboard, Web pages are represented with *keyword vector*s which are similar to *term vector*s used for the vector space model[13]. A keyword vector V_p of a Web page p described in the following.

$$\begin{aligned} V_p &= (v_{p1}, v_{p2}, \cdots, v_{pn}) \\ &= \left(\frac{f(p,t_1)}{m(D,t_1)}, \frac{f(p,t_2)}{m(D,t_2)}, \cdots, \frac{f(p,t_n)}{m(D,t_n)} \right) \\ & \quad \text{where } \ m(D,t) = \max_{p \in D} f(p,t) \end{aligned}$$

The value v_{pi} stands for normalized occurrence frequency of the input keyword t_i in a Web page p, and shows the importance of t_i in the Web page. The $\max(t_i)$ of a keyword t_i is obtained by the following procedure. A keyword t_i is inputted to a search engine and the most relevant Web page to t_i is fetched. The occurrence frequency of t_i in the Web pages is $\max(t_i)$. Thus $\max(t_i)$ means a maximum occurrence frequency of t_i over a large number of Web pages. Since the $\max(t_i)$ is computed once for each keyword and fixed, a system can decrease the computational cost for updating a density distribution blackboard. The order of the computation is linear: $O(n)$ where n is the number of stored keyword vectors in a density distribution blackboard. They are normalized by the maximum value in Web pages of a Web page database D, thus the range of values is [0, 1].

4.2 A density distribution blackboard

A blackboard system[4] provides a useful architecture to control a distributed multi-agent system. Agents read and write the information to layered shared memory, called a *blackboard*, and share them flexibly. Thus we use a blackboard for Web robots to share the information. The shared information is a *density distribution* which is the distribution of the number of keyword vectors in a Web page database over the keyword vector space.

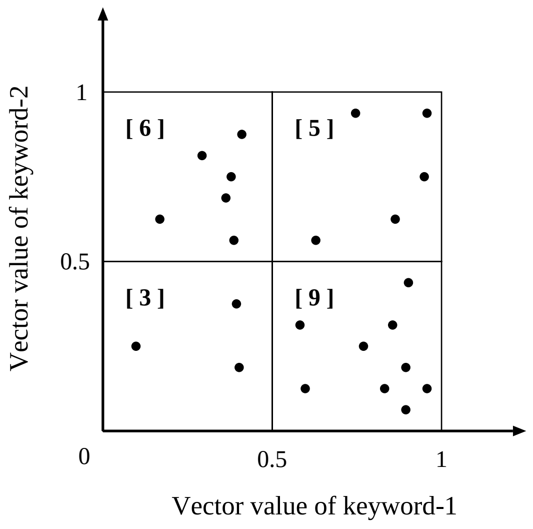

Figure 3. Density distribution blackboard.

The range [0, 1] of each axis in a n-keyword vector space is divided into m patches. Since the n-keyword vector space has n dimensions, m^n patches are generated in total. Next the densities for each patches are computed, and they are stored in a density distribution blackboard. Fig.3 shows a two dimensional density distribution blackboard for two keywords. Each axis is divided into two patches. The black points and the numbers within blankets indicate keyword vectors of Web pages and densities of patches respectively. If a patch has high density, it has been explored well and Web robots do not need to investigate there. If a patch has low density, it was not explored sufficiently and Web robots should investigate there more.

4.3 Anytime-control of Web robots using a blackboard model

4.3.1 Control policy for PWM

Though many search engines use multiple Web robots for gathering Web pages, few ones control them effectively. Thus we propose *anytime-control*: a novel method for controlling Web robots to build a PWM.

An important point to build a PWM interactively is that the building process may be interrupted anytime. A PWM system allows a user to interrupt information gathering to see intermediate results and control it. Hence it should be able to indicate *appropriate results* whenever a user requests. We consider the appropriate results are Web pages which were gathered *uniformly* on keywords. If it does not control Web robots, the obtained PWM will be uneven and its coverage will be far more narrow than an uniform one.

Fig.4 shows an uneven PWM and an uniform PWM

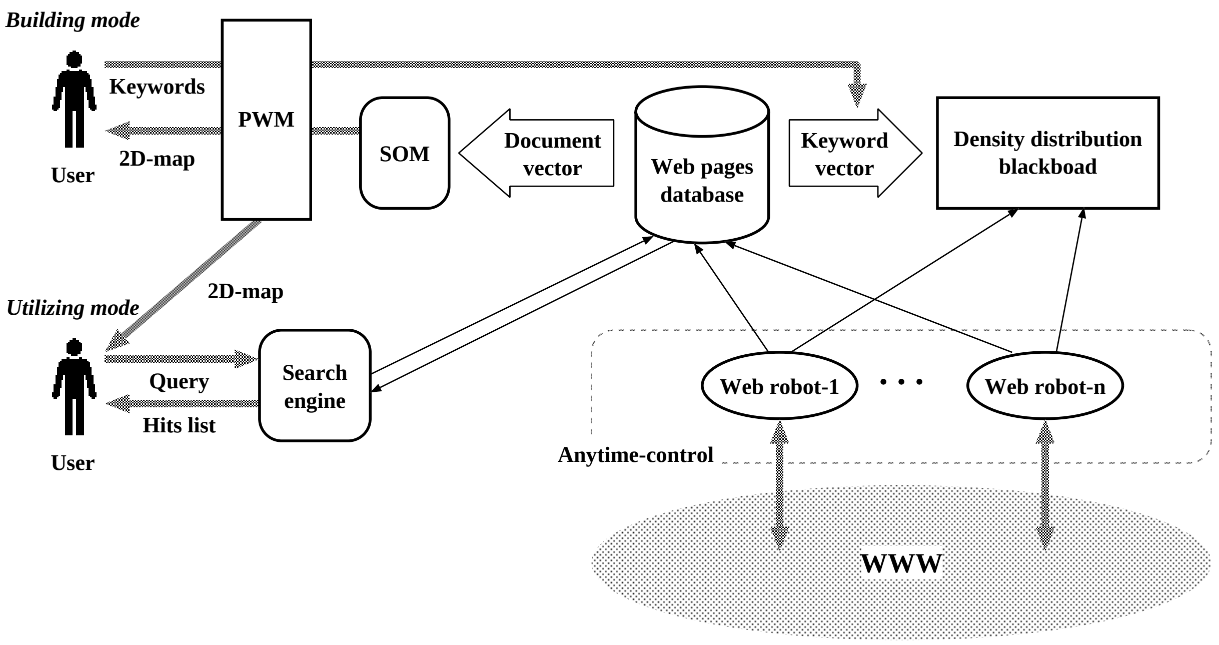

Figure 2. Overview of a PWM system.

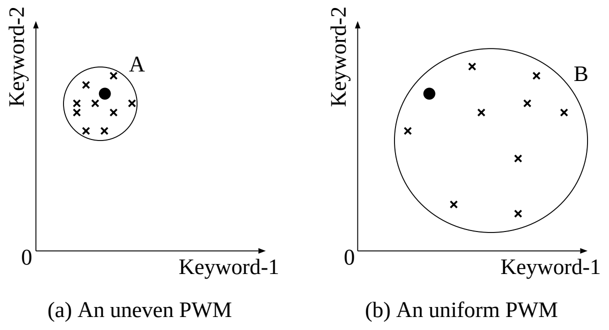

Figure 4. An uniform PWM.

when ten Web pages were gathered. The two keywords were given to a PWM system, and the *X* and *Y* axis indicate the relevance to keyword-1 and keyword-2 respectively. The black circles stand for start Web pages, and the crosses indicate keyword vectors of gathered Web pages. Though both of the PWM started from the same keyword vector, the results are different in the coverage (the circle A and B). Since a user selects a keyword vector about which he/she wants to gather more detail information, the coverage of subsequent layers is restricted within circle A and B. Thus the coverage of an uneven PWM becomes far more narrow than that of the uniform one. Obviously we prefer a PWM with a wide coverage.

4.3.2 A procedure for a Web robot

Thus we need an algorithm that returns an uniform PWM whenever a user interrupts. We developed such an algorithm which is called *anytime-control*[1]. The control is a simple distributed procedure shown in the following. Each Web robot acts using the algorithm asynchronously, and stops when a user interrupts it or the number of obtained Web pages becomes the given a limit number or a limit time has come.

Anytime-control procedure for a Web robot

1. See a density distribution blackboard and determine the most sparse patch.
2. Select a Web page α randomly in the patch.
3. Select a linked Web page β randomly in the Web page α.
4. Fetch a html file of the page β.
5. Generate a keyword vector from the obtained file and write it into a density distribution blackboard. Also store the file in a Web page database.

In the above procedure, Step 1 makes a PWM uniform. Each Web robot tends to explore in an area with the least

[1]This name is inspired by an anytime-algorithm[1] which, returns valid results anytime, has been studied in real-time problem solving in artificial intelligence.

information, and this keeps the density of Web pages uniform. Additionally, URLs of the fetched Web pages are stored, and the Web pages are not selected in Step 2 and 3.

Note that in order to gather Web pages in the same patch to the Web page α for Step 3 and 4, a system tries to get Web pages linked from the Web page α. This is a valid way supported by some evidences. Menczer[10] pointed out $R > G$, where R is the conditional probability that a Web page is relevant given that it is linked by another Web page that is relevant with the same query, and G is the generality of the query, i.e., the fraction of Web pages that are relevant. Since $R > G$ means that is is more likely to hit a relevant Web page from another relevant Web page than from any random Web page, the linked Web pages in Step 3 and 4 tend to be within the same patch to the Web page α. Another evidence is a real-time search mode of WebCrawler[3]. The algorithm assumes that following links from Web pages that are similar to what the user wants is more likely to lead to relevant Web pages than following any link from any Web page. Under the assumption, WebCrawler works well.

4.3.3 A procedure of a PWM system

Eventually the procedure of a PWM system is described in the following.

1. Initialize a WPS_{00} as a empty set, and set *Current* WPS_{ij} by WPS_{00}.
2. A system inputs MetaCrawler[11] the keywords as a query, and obtains the most relevant 10 Web page. A PWM system initializes a density distribution blackboard by giving it the document vectors of the 10 Web page as a start point.
3. Start Web robots under anytime-control for gathering relevant Web pages.
4. If interrupt of a user is given, work SOM to make a 2D-map of PWM and display it to a user. If a user clicks a node in the 2D-map, set WPS_{ij} by $i = i+1$, and go to Step 2.
5. If the number of gathered Web pages becomes the limit number, stop and wait a user's request.

5 Human computer interaction

5.1 SOM-based display

SOM (Self-Organizing Maps)[8] have been applied to build a 2D-map from a large text database. Because the learning algorithm is simple, and the input of high-dimensional vectors are automatically classified. For example, WEBSOM[5][6] has been developed for clustering a lot of articles for network news.

Since the keyword vector has the identical dimension to the number of keywords and it is usually more than three, a system needs to reduce the dimension to two in order to indicate a PWM to a user. Thus we can construct a 2D-map of a PWM by using SOM. We explain the SOM-based procedure in the following. SOM learns with random input vectors in the document vector space, and the document vectors of gathered Web pages are given to the learned SOM and classified.

1. Generate a set S of *document vector*s from Web pages of current WPS in a Web page database. The document vector is slightly different from a keyword vector in normalization. They are normalized as a unit vector using the following formula. E_p and $f(p,t)$ are the document vector of a Web page p and the occurrence frequencies of a word t in a Web page p.

$$\begin{aligned} E_p &= (e_{p1}, e_{p2}, \cdots, e_{pn}) \\ &= \left(\frac{f(p,t_1)}{l(p)}, \frac{f(p,t_2)}{l(p)}, \cdots, \frac{f(p,t_n)}{l(p)}\right) \\ &\quad \text{where } l(p) = \sqrt{\sum_{i=1}^{n} f(p,t_i)^2} \end{aligned}$$

2. Fig.5 shows the structure of SOM. Construct SOM using the number of keywords as the input nodes and c as the number of competitive nodes. The competitive layer is set to two dimension.
3. Generate a set R of random document vectors with random values through [0, 1] in each dimension of a document vector. Give the R to SOM repeatedly, and update is done as the following[8].
   (a) Let an input vector and weights of links from all input nodes to a competitive node u_i be $E_p = [e_1, e_2, \cdots, e_n]$ and $U_i = [u_{i1}, u_{i2}, \cdots, u_{in}]$ respectively.
   (b) SOM compute the similarity between the input vector and competitive nodes. The similarity is evaluated by Euclidean distance. using th the following formula.

$$\|E_p - U_i\| = \sqrt{\sum_j (e_{pj} - u_{ij})^2}$$

   (c) Since we need a 2D-map, a square having the winner as the center is used as neighborhood. The formula for updating the weights of neighbors is shown in the following, where the α is a learning rate. The α and the size of neighborhood are scheduled to decrease as learning progresses.

$$u_{ij}^{\text{new}} = u_{ij}^{\text{old}} + \Delta u_{ij}$$

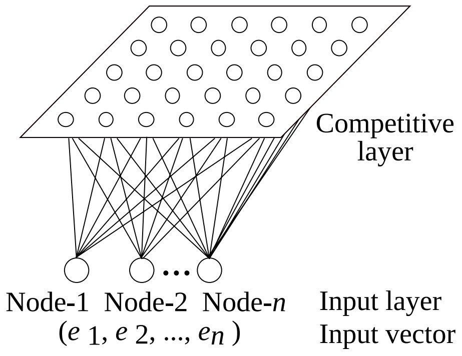

Figure 5. Self-Organizing Network.

$$\Delta u_{ij} = \begin{cases} \alpha(e_j - u_{ij}) & : i \text{ in the neighborhood} \\ 0 & : \text{otherwise} \end{cases}$$

4. After learning, the set S of document vectors of gathered Web pages is inputted to a learned SOM, winner nodes for each vector are determined, and classification is done. The winner node of a Web page indicates the class. Also *unit keyword vector*s are inputed to a learned SOM and the winner nodes are labeled with corresponding keywords. The winner nodes are called *keyword node*s, and the unit keyword vector of keyword n is $(0, \cdots, 1, 0, \cdots, 0)$ where only the nth value is 1. The keyword nodes are colored differently, and the other nodes among them are colored in graduation. Furthermore the size of a node is in proportion to the number of keyword vectors whose winner is the node.

We show an execution example in which a user wanted to know about "intelligence information retrieval in the WWW" and input five keywords: "intelligence", "retrieval", "gathering", "WWW" and "information". The limit number was set 1000 for anytime-control, and it took 16 hours for a PWM system to fetch the 1000 Web pages. The number of patches on each axis is ten, thus the amount of patches is 10^5 because the number of keywords is five. We use two Web robots for gathering Web pages.

After gathering Web pages, a learned SOM with random document vectors classifies the Web pages. We set the number c of competitive nodes by 100 which are positioned 10×10.

The number of training examples was set 10000, hence the 1000 document vectors of gather Web pages are inputted to SOM ten times. The initial learning rate and the initial neighborhood were set $\alpha = 0.5$ and a square with a 5 length sides (ds) respectively, and they were decreased linearly using following formulas where t is the number of training examples.

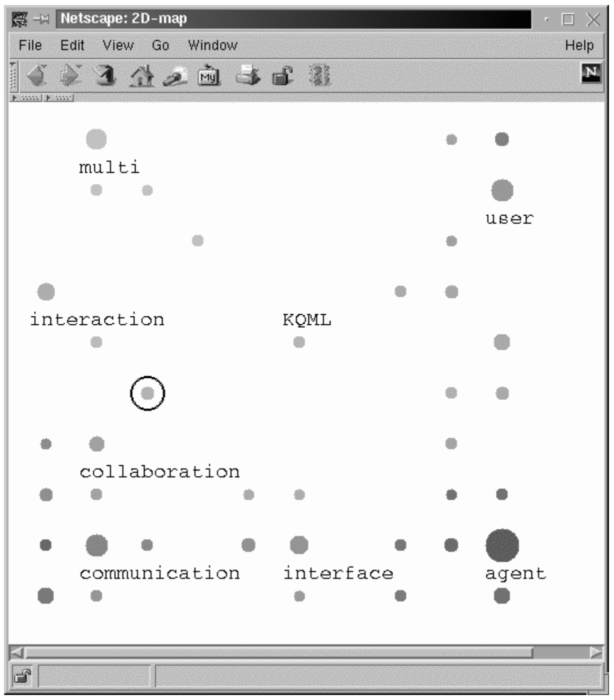

Figure 6. 2D-display of PWM.

$$\alpha = 0.5\left(1 - \frac{t}{10000}\right) \qquad d = 5\left(1 - \frac{t}{10000}\right)$$

An executed example of 2D-map for first layer PWMs is shown in Fig.6. The 2D-map was generated from keywords of subject-1 in Table. 1. You can see colored keyword nodes, and the distance between them stands for their similarity. The size of a node stands for the number of Web pages classified in the node.

5.2 User feedback

Using a 2D-map, a user can easily point nodes about which he/she wants to know more. When a node is pointed by a user, a system opens a dialog window called a *node info window* in which a user can see the titles of the Web pages whose winner node is the pointed node and the occurrence frequencies of all words in the Web page. A user determines whether he/she wants to know more about the node, and clicks the more detail button for request. Note that a 2D-map provides access to intermediate nodes except keyword nodes. This makes user's selection of intermediate concepts easy.

If the more detail button is clicked, a PWM system will gather Web pages for constructing the more detail *WPS* in the next deeper layer (Fig.1). The procedure is basically similar to one described in 4.3.2, however the density distribution blackboard is restricted. The occurrence frequencies of n keywords is restricted within the range $[\min_1, \max_1]$, $[\min_2, \max_2]$, $\cdots$, $[\min_n, \max_n]$, where $\min_i$ and $\max_i$ are the minimum value and the maximum value of ith keyword in the Web pages included in the pointed node. Thus a system gathers only the Web pages having keywords included in the restricted area. This restricts the next WPS within the neighborhood of the pointed node.

5.3 Search engine

A PWM system also provides a search engine on a Web page database to a user. The search engine is built by using a full-text search method straightforward. Unfortunately this search engine is not fully implemented at present.

5.4 Evaluation

5.4.1 Evaluating anytime-control

In order to verify the effect of anytime-control, we make experiments. To compare with anytime-control, we use general *Web robot search* used in many search engines. In general, Web robots are not control effectively and the strategy is a simple breadth first search. Thus we call it *Web robot search* described in the following.

Web robot search

1. Initialize the Web page list F = [ ].
2. Set a current Web page α by a start Web page, and append all linked pages in α to F.
3. Pick the head page β in F and remove it.
4. Fetch the page β.
5. Add all linked pages in β into the tail of F.
6. Go to Step 3.

For evaluating the effect of anytime-control, we investigate the standard deviation SD of the number of Web pages in each patch, which is describing like the following formula. x_i and $\bar{x}$ are the number of Web pages in patch i and the average of them over all the patches. n is the total of Web pages. The SD indicates the scattering of Web pages, thus the PWM is more uniform as the SD is smaller.

$$SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Fig.7 shows the experimental results. In the graph, the x-axis and y-axis stand for the number of gathered Web pages and SD. As seeing from the graph, anytime-control is able to keep a PWM more uniform than Web robot search.

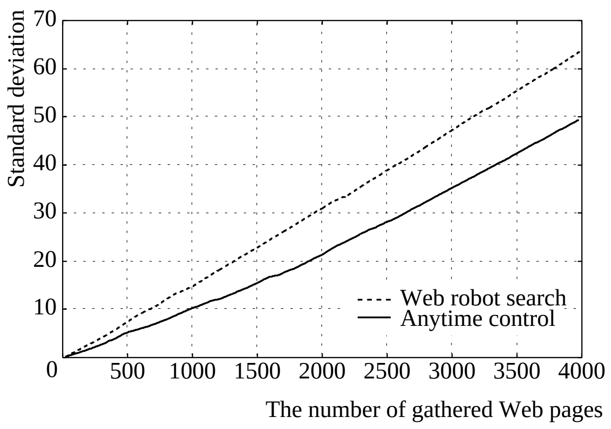

Figure 7. Uniformity of anytime-control.

5.4.2 Evaluating the gathered relevant Web pages

The purpose of a PWM system is to gather relevant Web pages to a user's interest effectively. Thus we make experiments with 8 subjects for evaluating Web pages gathered by a PWM system. The subjects are Master course students in a computer science department. Two methods: a PWM system with anytime-search and that with Web robot search are compared through the following identical procedures in same settings. Then we investigate the number of relevant Web pages in gathered Web pages in the 1st and 2nd layer WPSs. We implemented a PWM system using Perl, Java, SQL and parser programs on Linux in a Pentium II (233M Hz) PC-AT machine with 128M RAM. Four Web robots works on the machine as different processes, and they are not in parallel .

First the subjects independently input 5∼7 keywords (partially shown in Table. 1) to a PWM system and a system gathers relevant Web pages for three hours using anytime-control or Web robot search. Then first layer PWMs are generated. The number of patches on each axis is five, thus the amount of patches is 5^n where the number of keywords is n. A system generates the 2D-map of first layer PWM and indicates it to a user, and he/she selects a node about which he/she wants to know more. Next a system gathers the second layer PWM for three hours, and stops.

For 8 subjects, we investigate the average number of relevant Web pages in the first, second and both layer PWM generated by anytime-search and Web robot search. The relevance is determined by subjects. The experimental results are shown in Fig.8 where the *average of ratio* $= \frac{The\ number\ of\ relevant\ pages\ gathered\ by\ anytime-search}{The\ number\ of\ relevant\ pages\ gathered\ by\ Web\ robot\ search}$ for first layer, second layer and both of the layers is indicated. When the ratio is more than one, anytime-search gathered more relevant Web pages than Web robot search. As seeing from this figure, our PWM system using anytime-search outper-

Table 1. Keywords given by subjects (partially).

Subject-1	KQML, multi agent, communication, collaboration, interaction, interface, user
Subject-2	GP, khepera, Genetic Programming, robot, mobile, service, intelligence
Subject-3	agent, www, navigator, personalize, profile, AI

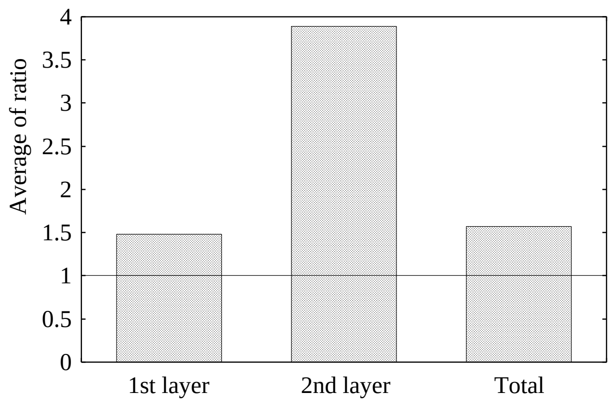

Figure 8. The average ratio of relevant Web pages.

forms that using Web robot search in all cases, and we find out our approach is promising to gather relevant Web pages effectively. Specially the 2nd layer shows the advantage of anytime-search.

6 Conclusion and open problems

We proposed PWM (Personal Web Map) for a user to gather interesting Web pages efficiently, and developed a PWM system. The system is able to anytime-control for multiple Web robots to gather relevant Web pages effectively. For controlling Web robots, a density distribution blackboard is used, and an uniform distributed PWM is built. From Web pages in the database, document vectors are generated and used to SOM-based classification for constructing 2D-map. A user easily recognizes PWM through the 2D-map, and gives feedback by pointing a node about which he/she wants more detail information. We also implemented a PWM system and made experiments for evaluation. As results, we found out that our PWM is a promising approach to assist a user in gathering relevant information in the WWW.

References

[1] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 979–984, 1989.

[2] P. D. Bra and R. Post. Information retrieval in the world-wide web: Making client-based searching feasible. In *the First International WWW Conference*, 1994.

[3] F. Cheong. *Internet Agents: Spiders, Wanderers, Brokers, and Bots*. New Riders, 1996.

[4] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II speech-understanding system: Integrated knowledge to resolve uncertainty. *Computer Surveys*, 12:213–253, 1980.

[5] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. Newsgroup exploration with WEBSOM method and browsing interface. Technical Report A32, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[6] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[7] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 770–775, 1997.

[8] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, pages 1464–1480, 1990.

[9] H. Lieberman. Letizia: A agent that assists Web browsing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 924–929, 1995.

[10] F. Menczer. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 227–235, 1997.

[11] MetaCrawler. http://www.metacrawler.com/.

[12] R. C. Miller and K. Bharatb. Sphinx: a framework for creating personal, site-specific web crawlers. *Computer Networks and ISDN Systems*, 30(1–7):FP12, 1998.

[13] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.