A Dynamic Access Planning Method for Information Mediator*

Yasuhiko Kitamura, Tomoya Noda, and Shoji Tatsumi

Department of Information and Communication Engineering
Faculty of Engineering, Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan
{kitamura, tnoda, tatsumi}@kdel.info.eng.osaka-cu.ac.jp
http://www.kdel.info.eng.osaka-cu.ac.jp/~kitamura/

Abstract. The Internet is spreading into our society rapidly and deeply and is becoming one of our social infrastructures. Especially, WWW technologies are widely used for doing business and research, creating communities, disseminating personal information, and so on. We usually access WWW pages one by one through a browser, but we can add more value to them by integrating information collected from various WWW sites. However, to realize such WWW information integration, we face obstacles such as distributed information sources, access cost, and frequently and asynchronous updates of information. We here adopt mediator, which integrates information from distributed information sources, with cache mechanism to reduce access cost. We also propose a dynamic access planning method to cope with frequently updating information sources. In a limited time period, it can construct an appropriate answer by accessing information sources effectively considering reliability and quality of cached data. We show its performance through a real-world flight information service comparing with a conventional access strategy.

1 Introduction

The Internet is spreading into our society rapidly and deeply and is becoming one of our social infrastructures which are indispensable for our daily life. Especially, WWW technologies are widely used for doing business such as electronic commerce, doing research, creating special interest communities, disseminating personal information, and so on. We usually use the WWW by accessing WWW pages one by one through a browser, but we can add more value to it by integrating information collected from various WWW sites. Of course, we can make a link page or a portal site where related WWW pages are hyperlinked. Search engine is another elaborated approach for information integration, which dynamically generates a list of WWW pages for specified keywords. However, these approaches provide only a link set of related WWW pages and leave users collecting the pages, extracting data from them, and integrating the data to

* This work is partly supported by NTT Communication Science Laboratories.

obtain desired information. To reduce this bothering work, WWW information integration [7] aims at providing a way to collect, extract, and integrate information from various WWW sites flexibly and automatically.

A typical example of WWW information integration is flight information service. In Japan, each of major airline companies provides a flight information service which we can consult about flight schedule and availability through the WWW. To our data input about departing date, origin, and destination, it returns a list of flight number, schedule, and availability. However, in case several airline companies operate on common routes, we need to access each of the WWW sites respectively to find connecting flights over different airline companies. WWW information integration service can provide a comprehensive and unified view of flight information over multiple information sites by collecting and integrating the information on behalf of users.

However, we have three major issues to tackle for achieving WWW information integration as follows.

(1) Distributed Autonomous Information Sources: WWW information sources are distributed on the Internet and maintained individually. Hence, we need to collect related information from distributed information sources.

(2) Access Cost: It takes time and cost to access information sources through the Internet. Moreover, WWW information integration often needs to collect a large amount of data, so it takes much time and cost.

(3) Frequently and Asynchronous Updates: Some information sources are frequently and asynchronously updated. The timing of update depends on the source, so we cannot know when it is updated until we actually access it.

To cope with (1), we adopt mediator [15] which integrates distributed information as shown in Fig. 1. When a mediator receives a query from a user, it accesses multiple WWW sites to collect WWW pages. It then extracts data from the collected pages to construct an answer to the user's query. To cope with (2), we can cache collected data from WWW sites, so we can improve the response time to users because we can reduce the number of accessing WWW sites.

However, as shown in (3), WWW sites are frequently updated, so cached data may well be obsolete shortly and may lead to construct an incorrect answer. On the other hand, if we do not use a cache mechanism, we need to take a long time to collect a large amount of data. Hence, how to collect data properly in a limited time from frequently updated WWW sites becomes an important research issue. In this paper, we propose a dynamic access planning method for information mediator which collects and caches data to construct an answer to user's query. It makes an access plan dynamically considering reliability and quality of cached data to construct a proper answer in a limited time.

In Section 2, we propose a dynamic access planning method, and show its performance by applying it to a real-world flight information service in Section 3. We discuss related work in Section 4 and conclude our discussion in Section 5.

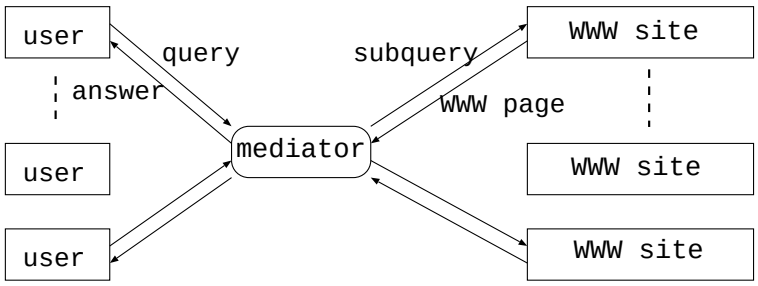

Fig. 1. Information integration through a mediator.

2 Dynamic Access Planning for Information Mediator

We use a mediator to integrate information from multiple WWW sites. When a mediator receives a query from a user, it collects pages from multiple WWW sites, extracts and integrates required data from them to construct an answer. To reduce the number of access, it can cache the collected data. However, many WWW sites are often frequently updated, so it needs to reload WWW pages to update cached data. On the other hand, the mediator is obliged to return an answer to the user within an allowable time period, so it needs to select pages to reload because it may take a long time, which may be intolerable for the user, to update the whole cache. However, partially updated cached data may lead to construct an incorrect answer. Hence, how to select pages to reload is an important issue to construct an appropriate answer in a limited time. We here propose a dynamic access planning method which considers reliability and quality of cached data. Within an allowable time period, this method repeats to select and reload WWW pages to update the cached data, then construct an answer.

At first, we define several terms such as facts, solutions, reliability, and quality, used in this paper.

2.1 Facts

A mediator collects WWW pages and extracts data from them. We call an atomic piece of extracted data *fact*. A number of facts can be extracted from a single WWW page. In a flight information service with flights like Fig. 2 for example, flight schedule and availability can be extracted from WWW sites of airline companies as facts as shown in Fig 3. In this example, `flight(JAL124,Sapporo,Itami, 0800,0900)` means that JAL124 departs from Sapporo at 8AM and arrives at Itami at 9AM, and `availability(JAL124,1999/12/16,Yes)` means that JAL124 is available on December 16th, 1999. Facts are stored in mediator's cache once they are collected.

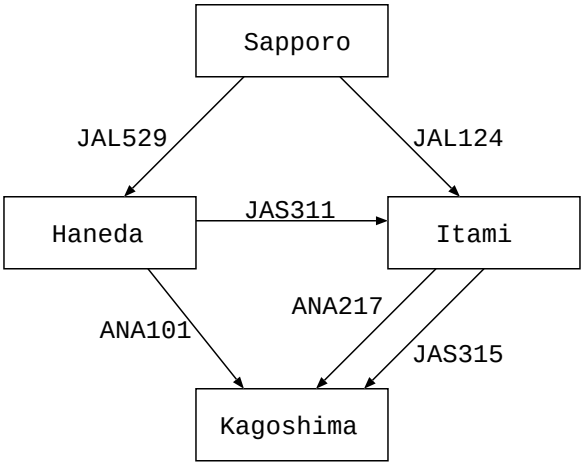

Fig. 2. Flight connection.

2.2 Reliability of Facts

Because some WWW sites are frequently updated, the difference between cached data and original data becomes large as time goes by. We define the *reliability* of a fact f as a function $r(f)$ which takes a real value between 0 and 1. The reliability function returns largest (1) when the fact is just loaded and monotonically decreases until 0 as time elapses.

The frequency of updates, or the shape of the reliability function, depends on the fact and the WWW site. For example, flight availability is frequently updated, so its reliability decreases soon. On the other hand, flight schedule is moderately updated once a month or so, so the reliability decreases slowly.

Though the shape of reliability function may differ depending on the type of fact, approximately we can use the following function,

$$r(f) = \frac{1}{1 + wt}$$

where w is a weight which depends on the type of fact and t is the elapsed time since f is updated. When w is large, the reliability decreases rapidly, and when w is small, it decreases slowly.

In this paper, we classify facts into two classes; *dynamic facts* and *static facts*. Dynamic facts are ones which may be updated and static facts are ones which may not. Generally speaking, we seldom have static or unchangeable facts in the real world, but in a short time span we can view some facts as static. For example, in a flight information service, flight availability can be viewed as a dynamic fact because it may be updated hourly or so, but flight schedule can be viewed as a static fact because it is updated in a longer time interval, monthly or so, than availability. Once a static fact is loaded and cached, it does not need to be reloaded.

Fact	Reliability
flight(JAL124,Sapporo,Itami,0800,0900)	1
flight(JAL529,Sapporo,Haneda,0700,0800)	1
flight(ANA217,Itami,Kagoshima,1000,1200)	1
flight(JAS315,Itami,Kagoshima,1300,1500)	1
flight(JAS311,Haneda,Itami,0830,0930)	1
flight(ANA101,Narita,Kagoshima,0930,1100)	1
availability(JAL124,1999/12/16,Yes)	0.90
availability(ANA217,1999/12/16,Yes)	0.60
availability(JAS315,1999/12/16,No)	0.30
availability(JAL529,1999/12/16,Yes)	0.70
availability(JAS311,1999/12/16,Yes)	1.00
availability(ANA101,1999/12/16,No)	0.60

Fig. 3. Facts for flight information service.

2.3 Answer

When a mediator receives a query from a user, it constructs an *answer* by using cached facts. How to construct an answer can be represented by Prolog-like rules as follows.

$$(R1)\ \texttt{query}(\$\texttt{A}, \$\texttt{B}) :- \$\texttt{fact}(\$\texttt{A}, \$\texttt{B}).$$
$$(R2)\ \texttt{query}(\$\texttt{A}, \$\texttt{B}) :- \$\texttt{fact}(\$\texttt{A}, \$\texttt{C}), \texttt{query}(\$\texttt{C}, \$\texttt{B}).$$

`$fact($A,$B)` represents a cached fact and `$A` and `$B` are variables. By using (R1), we can construct an answer directly from a single fact, and by using both of (R1) and (R2), we can construct an answer from multiple facts. Hence, an answer can be viewed as a set of facts.

For example, some rules for flight information service are given as follows.

$$\begin{aligned}(R3)\ &\texttt{query}(\$\texttt{P1}, \$\texttt{P2}, \$\texttt{Date}, \$\texttt{T1}, \$\texttt{T2}) :- \\ &\quad \$\texttt{flight}(\$\texttt{Number}, \$\texttt{P1}, \$\texttt{P2}, \$\texttt{Dep}, \$\texttt{Arr}), \\ &\quad \$\texttt{availability}(\$\texttt{Number}, \$\texttt{Date}, \$\texttt{Status}), \\ &\quad \$\texttt{Dep} >= \$\texttt{T1}, \\ &\quad \$\texttt{T2} >= \$\texttt{Arr}.\end{aligned}$$
$$\begin{aligned}(R4)\ &\texttt{query}(\$\texttt{P1}, \$\texttt{P2}, \$\texttt{Date}, \$\texttt{T1}, \$\texttt{T2}) :- \\ &\quad \$\texttt{flight}(\$\texttt{Number}, \$\texttt{P1}, \$\texttt{P3}, \$\texttt{Dep}, \$\texttt{Arr}), \\ &\quad \$\texttt{availability}(\$\texttt{Number}, \$\texttt{Date}, \$\texttt{Status}), \\ &\quad \$\texttt{Dep} >= \$\texttt{T1}, \\ &\quad \$\texttt{T2} >= \$\texttt{Arr}, \\ &\quad \texttt{query}(\$\texttt{P3}, \$\texttt{P2}, \$\texttt{Date}, \$\texttt{Arr}, \$\texttt{T2}).\end{aligned}$$

(R3) represents a query to find a direct flight on date `$Date`, which departs from airport `$P1` after time `$T1` and which arrives at airport `$P2` by time `$T2`. (R4) represents a query to find connecting flights through airport `$P3`.

For example, when a query `query(Sapporo,Kagoshima, 1999/12/16,0600,1600)` to find a route from Sapporo to Kagoshima is given, an answer can be constructed as a set of facts from cached facts in Fig. 3 as follows.

```
flight(JAL124, Sapporo, Itami, 0800, 0900)
availability(JAL124, 1999/12/16, Yes)
flight(ANA217, Itami, Kagoshima, 1000, 1200)
availability(ANA217, 1999/12/16, Yes)
```

The complete set of answers to the above query is shown in Fig. 4.

Answer	Facts
A_1	`flight(JAL124,Sapporo,Itami,0800,0900), seat(JAL124,1999/12/16,Yes),` `flight(ANA217,Itami,Kagoshima,1000,1200), seat(ANA217,1999/12/16,Yes).`
A_2	`flight(JAL124,Sapporo,Itami,0800,0900), seat(JAL124,1999/12/16,Yes),` `flight(JAS315,Itami,Kagoshima,1300,1500), seat(JAS315,1999/12/16,No).`

A_3	`flight(JAS529,Sapporo,Haneda,0700,0800), seat(JAS529,1999/12/16,Yes),`
	`flight(JAS311,Haneda,Itami,0830,0930), seat(JAS311,1999/12/16,No),`
	`flight(ANA217,Itami,Kagoshima,1000,1200), seat(ANA217,1999/12/16,Yes).`

A_4	`flight(JAS529,Sapporo,Haneda,0700,0800), seat(JAS529,1999/12/16,Yes),`
	`flight(JAS311,Haneda,Itami,0830,0930), seat(JAS311,1999/12/16,No),`
	`flight(JAS315,Itami,Kagoshima,1300,1500), seat(JAS315,1999/12/16,No).`

A_5	`flight(JAS529,Sapporo,Haneda,0700,0800), seat(JAS529,1999/12/16,Yes),` `flight(ANA101,Narita,Kagoshima,0930,1100), seat(ANA101,1999/12/16,Yes).`

Fig. 4. A complete set of answers to query `query(Sapporo,Kagoshima, 1999/12/16,0000,2359)`

2.4 Quality of Answer

Quality of answer represents how an answer satisfies user's requirement and is used for selecting facts to reload and for ranking answers. Quality of answer is calculated from quality of facts which compose the answer. We can divide quality of answer into *dynamic quality* and *static quality*. The dynamic quality concerns only dynamic facts and the static quality concerns only static facts.

How to calculate dynamic or static quality of answer depends on the service which the mediator provides. For example, in a flight information service, we

can calculate quality of answer from flight schedule and availability. If an answer provides flights with which the traveling time is as short as possible and which are available, then its quality is high. In this example, the static quality concerns flight schedule, so we can define static quality of answer A as

$$Q_S(A) = \begin{cases} 1 - \dfrac{t_a - t_d}{24} & 0 \leq t_a - t_d \leq 24, \\ 0 & \text{otherwise}, \end{cases}$$

where t_a is the desired arrival time and t_d is the departure time of the first flight. We assume the quality function returns a real value between 0 (worst) and 1 (best). The dynamic quality concerns flight availability, so we can define it as

$$Q_D(A) = \begin{cases} 1 & \text{if every flight is available}, \\ 0 & \text{otherwise}. \end{cases}$$

For example, as shown in Fig. 4, there are five alternatives from Sapporo to Kagoshima and their arrivals are at 11:00, 12:00, or 15:00. If a user wants to arrive there by 16:00, the static quality of A_1 is

$$Q_S(A_1) = 1 - \frac{16 - 8}{24} = 0.67.$$

The dynamic quality of A_1 is 1 because both of JAL124 and ANA217 are available. The static quality of A_2 is identical with that of A_1 but its dynamic quality is 0 because JAS315 is not available.

We here define the total quality of answer by multiplying static quality and dynamic quality as

$$Q(A) = Q_S(A) \cdot Q_D(A).$$

2.5 Reliability of Answer

An answer is composed of one or more facts. We can define *reliability of answer* from reliability of composed facts. We here define reliability of answer to be the minimum reliability among composed facts as

$$R(A) = \min_{f \in A} r(f).$$

For example, the reliability of answer A_1 is

$$R(A_1) = \min\{1, 0.90, 1, 0.60\} = 0.60.$$

2.6 Dynamic Access Planning Algorithm

A mediator constructs an answer from cached facts, but the reliability of answer decreases as the reliability of composed facts decreases. To increase the reliability of answer, we need to update the facts by reloading WWW pages, but should finish it within an allowable time period which is tolerable for the user. Hence,

1: construct answers $\{A_1, A_2, \cdots\}$ to a user's query by using cached facts
2: repeat until query time expires {
3: select A_i where $S(A_i) = \max\{S(A_1), S(A_2), \cdots\}$
4: select f_m where $r(f_m) = \min_{f_n \in A_i} r(f_n)$
5: reload a WWW page and update f_m }
6: sort answers by $P(\cdot)$
7: return the best answer to the user

Fig. 5. Dynamic access planning algorithm.

for obtaining a good answer in a limited time, it is important how to select facts to update. Here we define two scores $S(A)$ and $P(A)$ for an answer A. $S(A)$ is used to select facts to update and $P(A)$ is used to rank the answers. We show our dynamic access planning algorithm in Fig. 5.

When a mediator receives a query from a user, it constructs answers by using rules and cached facts (Line 1).[1] Then, it repeats to update facts within an allowable time period (Lines 2 to 5). In the loop, it selects a fact to update (Line 3). At first, it calculates $S(A_i)$ for each A_i as

$$S(A_i) = Q_S(A_i) \times (1 - R(A_i)).$$

It selects an answer with high static quality and low reliability because it may well lead to a good answer with high reliability if it is updated. We need not consider answers with low quality because it does not lead to a good answer even if they are updated nor ones with high reliability because we need not update them. We here consider only static quality because dynamic quality is unknown until the fact is updated. It then selects a fact with the lowest reliability from the selected answer (Line 4). Finally, the mediator reloads a page from a WWW site to update the fact (Line 5).

When the updating time expires, the mediator ranks answers by $P(A_i)$ to show the best one to the user. $P(A_i)$ is defined as

$$P(A_i) = Q(A_i) \times R(A_i)$$

which reflects quality and reliability of answers (Lines 6 and 7).

Here we show an example of how facts are selected to be updated by using Table 1. At first, answer A_2 is selected because its S score is best among 5 candidates. A_2 has an unavailable flight JAS315 at this moment (Fig. 4), but the fact may have been updated because the reliability is low (0.30) as shown in Fig. 3. Hence, fact `availability(JAS315,1999/12/16,No)` is selected to be updated. Let us assume it becomes `availability(JAS315,1999/12/16,Yes)` with its reliability of 1. Then, answers A_2 and A_4 are updated because they include the above fact. Now, the best S score is with A_1, so A_1 is selected and

[1] We assume that all the facts have been cached, the reliability of static facts and that of dynamic facts are initially set to be 1 and 0 respectively.

a fact `availability(ANA217,1999/12/16,Yes)` is updated. Now let us assume that it is updated to be unavailable, then Q values of A_1 and A_3 become 0. If the time for updating expires, A_2 is shown to the user as the best answer because it has the best P. Like above, until the time expires, facts to update are selected dynamically.

In this example, if only once of update is allowed, then answer A_2 with P score of 0.60 is selected for the query. If no update is allowed, A_1 with 0.40 is selected.

Table 1. Changes of function values according to the number of updates.

A_i	Q_s	No update				1st update				2nd update			
		Q	R	P	S	Q	R	P	S	Q	R	P	S
A_1	0.67	0.67	0.60	0.40	0.27	0.67	0.60	0.40	0.27	0.00	0.90	0.00	0.07
A_2	0.67	0.00	0.30	0.00	0.47	0.67	0.90	0.60	0.07	0.67	0.90	0.60	0.07
A_3	0.63	0.63	0.60	0.38	0.25	0.63	0.60	0.38	0.25	0.00	0.70	0.00	0.19
A_4	0.63	0.00	0.30	0.00	0.44	0.63	0.70	0.44	0.19	0.63	0.70	0.44	0.19
A_5	0.63	0.00	0.60	0.00	0.25	0.00	0.60	0.00	0.25	0.00	0.60	0.00	0.25

3 Evaluation Experiment

An excellent mediator can return an optimal answer in a short time, so we measure how much our Dynamic Access Planning (DAP) method generates an optimal answer within a fixed time period or a fixed number of updates in other words.

We compare the DAP method with the FIFO where it update a fact as the first-in first-out basis.[2] In other words, the FIFO method updates facts in order from the oldest one in the cache. FIFO is identical with DAP when

$$S(A) = 1 - R(A).$$

We performed an evaluation experiment by applying the methods to a flight information service which integrates data from airline WWW sites. We extracted facts about flight schedule and availability from three WWW sites which provide flight information run by JAL (Japan Air Line)[3], ANA (All Nippon Airways)[4], and JAS (Japan Air Systems)[5]. We regard flight schedule as static facts and

[2] The LRU (Least Recently Used) algorithm is another well known caching strategy, but it is not suitable for the comparison because it assumes static information sources.

[3] http://www.5971.jal.co.jp/cgi-bin/db2www/avail.d2w/report

[4] http://rps.ana.co.jp/drs/vacant1.cgi

[5] http://www.jas.co.jp/kusektop.htm

availability as dynamic facts, and we define quality and reliability of answer and fact as in Section 2.

We submitted a query to the mediator every hour from December 18th, 1999 to December 28th, 1999.[6] Queries are created automatically and each of them is to find flights on December 28th between two airports randomly chosen among Sapporo, Haneda, Itami, Kansai, Fukuoka, and Naha. We limited the number of connection to 1 to keep the size of problem small as we need to make it feasible in the real-world setting. The ratio of obtaining an optimal answer is shown in Fig. 6. The horizontal axis shows the number of updates, and the vertical axis shows the ratio of obtaining an optimal answer. Both of methods improve its performance as the number of updates increases. Roughly speaking, if the DAP accesses WWW sites about 8 times for a query, it can construct an optimal answer. On the other hand, the FIFO takes about 18 times. This is because the DAP selects facts to update more carefully than the FIFO as it does not consider only reliability of solution but also quality of that. In this experiment, the size of problem is kept small to make it feasible in the real-world setting. As we limited the number of transit to 1, so an answer can be constructed from only 2 static facts and 2 dynamic facts. Moreover, an original flight information service returns a set of facts to a single subquery, so it reduces the number of WWW access. (In the ANA service, we could extracted 22 facts from a page to a single subquery.) However, if we enlarge the size of problem, the superiority of DAP method against the FIFO will become more remarkable.

4 Related Work

Research on information integration has been widely performed in the fields of database and artificial intelligence [16, 7, 4, 10]. For example, the TSIMMIS project at Stanford University database group aims at flexibly integrating various and heterogeneous information sources on the Internet [2]. In this project, information sources are not integrated directly like a distributed databases, but they are integrated through a *mediator* which receives queries from users, sends subqueries to information sources, integrates answers, and sends back the final answers to the users [15, 17]. In this paper, we also adopt mediator for WWW information integration. As an approach from AI, Stanford University logic group proposes a *federated system* [6] where information agents and facilitators integrate information by using ACL (Agent Communication Language) [3].

Researchers on WWW information integration mainly discuss modeling information sources [11], information extraction [8] and gathering [5, 13, 9], information match-making [12] so far and they seldom assume information sources which are frequently updated except work by Adali et al [1]. Adali and his colleagues propose query caching and optimization method in distributed mediator systems. They discuss a query caching and optimization scheme which considers

[6] We submitted 264 ($= 24 \times 11$) queries in total but the mediator succeeded to construct some answer for 213 queries because of the original WWW sites' faults.

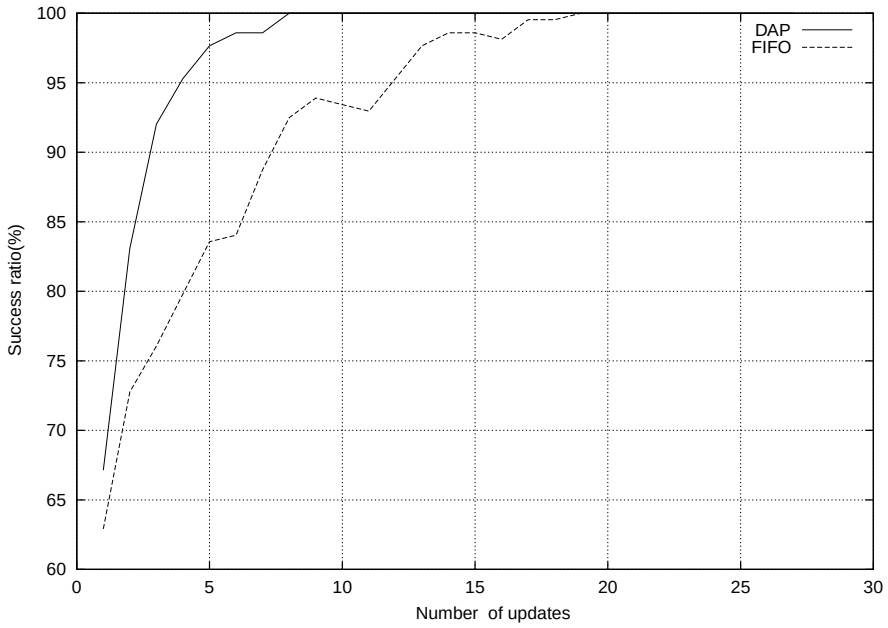

Fig. 6. Experimental result.

communication overhead, performance and faults of information sources, financial cost, and so on. In this paper, we discuss a dynamic access planning to obtain an appropriate answer in a limited time considering reliability and quality of answer but we do not have much interest in performance of information sources or communication channels. Work on view maintenance [18] tries to integrate information from multiple sources in a consistent manner, but we think keeping a large amount of information from distributed sources consistent raises communication overhead especially when information sources are frequently updated. In this paper, we take a semi-consistent approach by keeping cached information consistent as much as possible within an allowed amount of communication cost or time. Hence, our DAP algorithm has a characteristic of anytime algorithm [14] which returns some answer at anytime and which improves the quality as it takes more communication time.

5 Conclusion

We propose a dynamic access planning (DAP) method for mediators to integrate information from frequently updated WWW information sources. A mediator collects facts in a limited time properly and returns an appropriate answer to users. Our algorithm considers reliability and quality of answer to select facts to update. We show the superiority of the DAP method against conventional FIFO method by applying it to a real-world flight information service. In this paper,

we applied our method to only flight information domain, but we need to study the applicability and the feasibility of our method in other application domains such as portfolio management where information is frequently updated.

References

1. Adali, S., Candan, K.S., Papakonstantinou, Y., and Subrahmanian, V.S.: Query Caching and Optimization in Distributed Mediator Systems, SIGMOD-96 (1996) 137–148
2. Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J.: The TSIMMIS Project: Integration of Heterogeneous Information Sources. Proceedings of IPSJ Conference (1994) 7–18
3. Finin, T., Labrou, Y., and Mayfield J.: KQML as an Agent Communication Language. Bradshow, J.M. (ed.): Software Agents. AAAI Press (1997) 291–316
4. Florescu, D., Levy, A., and Mendelzon, A.: Database Techniques for the World-Wide Web: A Survey. SIGMOD Record, 27(3) (1998)
5. Friedman, M., Levy, A. and Millstein. T.: Navigational Plans for Data Integration. AAAI-99 (1999) 67–73
6. Genesereth, M.: An Agent-Based Framework for Interoperability. Bradshaw, J.M. (ed.): Software Agents. AAAI Press (1997) 315–345
7. Hearst, M.: Information Integration. IEEE Intelligent Systems 13(5) (1998) 12–24
8. Hsu, J.Y. and Yih, W.: Template-based Information Mining from HTML Documents. AAAI-97 (1997) 256–262
9. Kitamura, Y., Noda, T., and Tatsumi, S.: Single-agent and Multi-agent Approaches to WWW Information Integration. Ishida, T. (Ed.): Multiagent Platforms, Lecture Notes in Artificial Intelligence, Vol. 1599, Springer-Verlag. (1999) 133–147
10. Klusch, M. (Ed.): Intelligent Information Agents. Springer-Verlag (1999)
11. Knoblock, C.A., Minton, S., Ambite, J.L., Ashish, N., Modi, P.J., Muslea, I., Philpot, A.G., and Tejada, S.: Modeling Web Sources for Information Integration. AAAI-98 (1998) 211–218
12. Kuokka, D. and Harada, L.: Integrating Information via Matchmaking. Journal of Intelligent Information Systems 6 (1996) 261–279
13. Kwok, C.T. and Weld, D.S.: Planning to Gather Information. AAAI-96 (1996) 32–39
14. Russell, S.J. and Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Inc. (1995) 844
15. Wiederhold, G.: Mediators in the Architecture of Future Information Systems. IEEE Computer, 25(3) (1992) 38–49
16. Wiederhold, G. (Ed.): Intelligent Integration of Information. Kluwer Academic Publishers (1996)
17. Wiederhold, G. and Genesereth, M.: The Conceptual Basis for Mediation Services. IEEE Expert, 12(5) (1997) 38–47
18. Zhuge, Y., Garcia-Molina, H., Hammer, J., and Widom, J.: View Maintenance in a Warehousing Environment. SIGMOD-95 (1995).