FindFlow: Visual Interface for Information Search based on Intermediate Results

Tomoyuki Hansaki **Buntarou Shizuki** **Kazuo Misue** **Jiro Tanaka**

Graduate School of Systems and Information Engineering
University of Tsukuba,
Tennoudai 1-1-1, Tsukuba, Ibaraki 305-8571, Japan
Email: {hansaki, shizuki, misue, jiro}@iplab.cs.tsukuba.ac.jp

Abstract

FindFlow is a search interface that enables users to construct queries visually on the screen. Because the constructed queries show the process of the search, the user can take a step forward in the search while monitoring the process. The queries can be recombined freely, and previous search results can be reused. In addition, FindFlow shows results interactively for each operation. FindFlow is advantages for searches in which the user repeats trial and error many times.

Keywords: Visual Query, Information Search, Interactive Interface, Visual Interface.

1 Introduction

Searching for information, whether it is for renting an apartment, making a purchase, reserving a hotel room, or various other purposes, is something many of us do in our daily lives. To find the information we want, we have to construct the query appropriately. For example, in searching for an apartment, factors that we might include in the query are rent, location, age, size and distance to a train station. The search is then conducted by trial and error. If the results are not satisfactory, we change some conditions and repeat the search.

Many systems show only results that match the query completely. In many cases, although the result matches the query, we are not satisfied with what the search has returned, but we do not know which part of the query we should change to find what we want. We have to change the query partly or entirely through trial and error many times to find the needed result (Teeven, Alvarado, Karger & Ackerman 2004).

At other times, we may want to reuse a previous search, However, this is often not possible because a system may keep only the last result even if it is used many times. This means we have to construct the same query over and over.

When we repeat a search many times, the query often changes as we get new ideas about how to find what we are looking for. Often the system keeps returning a result that does not really satisfy our original request, but accept it anyway and give up the search.

In any search, the user has to formulate a query, input the query, execute the search, and monitor the results. We think the search system should support these steps to make the whole search process easier and faster.

We have developed a visual search interface called "FindFlow". FindFlow features a directed graph and supports query construction, trial and error, and reuse. FindFlow shows not only the final result but also the intermediate results, based on which the user can construct or modify queries. We expect that FindFlow will help users find what they the desired results, because queries can be constructed easily, quickly, and adequately.

2 Design

2.1 Support for Constructing a Query

The system should support query construction. A complicated text query is difficult to understand even if we have constructed it ourselves. The system should therefore provide a view of the query that the user can easily comprehend. The system should also provide functions for easy and fast query construction. The user has to repeat search steps many times during a search. The system should therefore offer support to reduce search time.

2.2 Support Trial and Error

Since the user repeats trial and error in a search, the system should support this process. When a result is unacceptable, the query has to be modified, but, on the basis of only the final result, it is difficult to determine exactly how the query should be changed. If the system shows how each condition in the query influences the results, the user will know how to obtain better results. The system should also support search with multiple queries at the same time. The user changes the query and searches many times, when the user considers how a certain query might return a good result and changes the wording of the query accordingly. If searches could be performed using multiple queries at the same time, it would not be necessary to repeat the search steps so many times.

The system should show the result interactively so that the user does not have to do execute a search after each operation, and can confirm the result immediately.

2.3 Support for Reuse

Sometimes, the user wants to reuse a previous search when starting a similar one. The system should therefore support the reuse of previous searches.

3 FindFlow

FindFlow is a visual search interface that allows users to construct queries visually using a dataflow diagram

(Fig. 1).

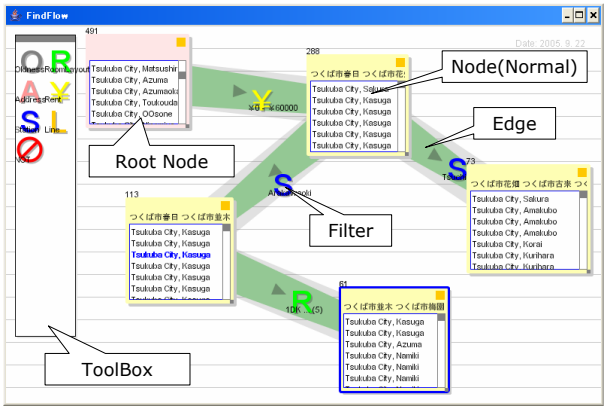

Figure 1: Screenshot of FindFlow: The user is searching for an apartment in Tsukuba.

Several visual interfaces that allow users to construct queries on the screen have been developed (Young & Shneiderman 1993, Guo 2003, Angelaccio, Catarci & Santucci 1990, Batini, Catarci, Costabile & Levialdi 1996, Catarci, Costabile & Levialdi 1993, Krishnan & Kunii 1991). We chose a visual interface based on a dataflow diagram because the whole process can be represented in one diagram, with makes things easy to understand, and because it is easy to compose queries with it (Tanimoto 2003).

FindFlow shows a visual query that describes both the query conditions and results, so that the user can know how the query should be changed. It also shows the results interactively after each user operation, and the user can construct the query while monitoring results.

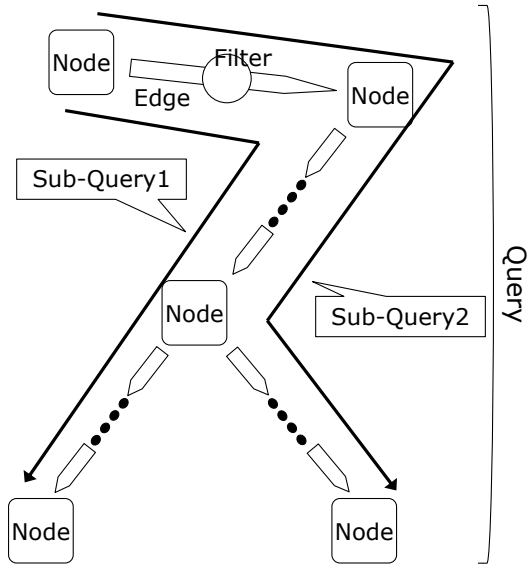

Figure 2: The query and a sub-query.

In FindFlow, a query is constructed using an edge, node, and filter. Various queries can be constructed at the same time (Fig. 2). The part of the query from the root node to another node is called a sub-query. The user can execute various searches at the same time using sub-queries.

3.1 Compositions for Constructing Query

Edge The edge connects two nodes and sends data from one node to the other.An arrow shows the direction the data is sent. A filter on the edge filters data from the sending node, and the thickness of the edge indicates how much data has been filtered. The thickness of the edge changes depending on the amount of data in both the sending node and the receiving node. This helps the user know how the filter works and determine how the query should be changed.

Node The node receives data from edges and shows the received data. The node shows headings for the data in a list and highlights frequently appearing words. The order of data shown in the list depends on the filters (described later). The user can know what kind of data the node contains and whether the filter is appropriate or not. If the filter is not appropriate, the user changes or rearranges the compositions. If the node is too small to see the result, the view can be enlarged by dragging from the right bottom corner of the node.

There are two types of nodes: a normal node, which is yellow, and a root node, which is red. The root node contains all data from the search. The user connects some normal nodes to the root node and starts the search.

Filter To help the user construct queries, the filter is shown as an icon on the screen. By dragging the icon, the user can add, remove, or exchange it anytime. The user can set up the filter on the edge by moving the icon to it. When the mouse pointer touches the edge as the icon in dragged, FindFlow interactively shows the results of applying the filter. This helps the user can determine how to construct a query while confirming the results. The filter shows the setting window when the icon is clicked. The user can decide how the filter will work in detail by changing the settings. Figure 3 shows a rent filter. The user can change the rent criteria by moving the scrollbar, which is shaded according to the amount of matched data. The scrollbar shows the results by shading, which supports trial and error.

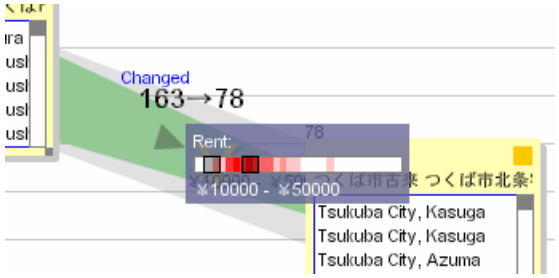

Figure 3: Setting window of rent filter

3.2 Operations

FindFlow's simple operations help make query construction easy, FindFlow shows the results interactively after each operation.

Creating a Node To create a node, the user simply clicks the right top button of a node.

Connecting Two Nodes Figure 4 shows the operation for making connections. To connect existing nodes, the user clicks the right upper button

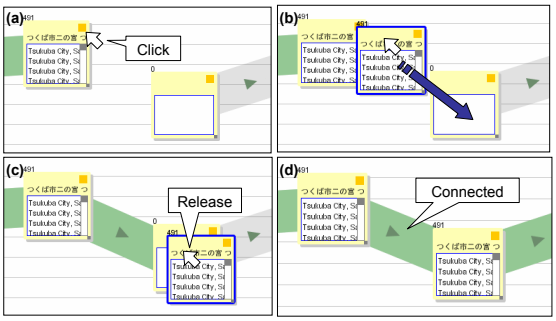

Figure 4: Operations for connecting nodes.

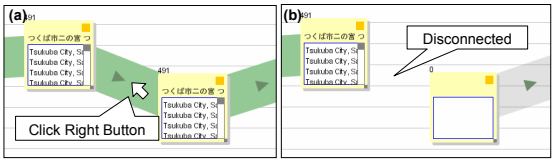

Figure 5: Operations for disconnecting nodes.

of the sending node (a). The new node is created (b), and the user drags it to the receiving node (c). FindFlow makes the edge work as the connection (d). When the connection is made, the sending node sends the data to the receiving node.

Disconnecting Two Nodes Figure 5 shows the operation for disconnecting nodes. To disconnect two nodes, the user moves the mouse pointer to the edge and clicks the right button (a). The connection between nodes disappears immediately, and no further data is transferred. (b). Any data in the receiving node is expunged.

Deleting a Composition In revising a query, the user may want to delete compositions. The operation for deleting composition is the same as the disconnecting operation . The user moves the pointer to the composition and right-clicks the mouse.

Making Divergences and Junctions The user can make divergences and junctions in a query. When setting up different compositions for a diverged query, the user can push different searches forward at the same time. This helps the user search with multiple queries at the same time. Figure 6 shows a divergence and junction. When the user makes a divergence, the divergence node sends the same data to each receiving node. When the user lets some queries combine, the junction node merges the data from each edge.

The operation for combining or diverging queries is the same as the one for making connections.

3.3 Weight of filters

The node shows lists the results and sorts the data based on the weight of placed filters on the query. The weight of the filter used near the root node increases, because it is reasonable to assume that users construct queries in the order they believe to be important.

By assigning a weight to filters, FindFlow supports the user in finding the needed data.

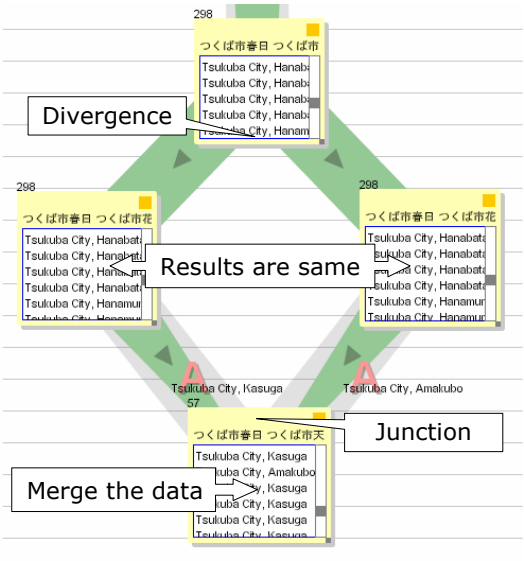

Figure 6: Divergence and junction

3.4 Step-by-Step Indication

FindFlow indicates how each filter influences the results in a step-by-step manner. When the user changes constructed query, the thickness of the edges changes and nodes change their result step by step from the changed composition to the end of query. This supports the user in finding how the operation influences the result, and makes it easy for the user to understand how data are filtered.

3.5 Packaging

FindFlow has a function for compiling sub-queries and generating new filters. This function helps the user in constructing the queries that will be used continually. Once it is made, the package can be reused many times. When a query becomes long, this helps the user in understanding the query.

Figure 7 shows the operation for packaging. To use the packaging function, the user drags the node to the toolbox and drops it (a-1). Then, a new package is generated as a filter(a-2), and the sub-query from the root node to the dropped node is compiled into the new package. The user names the generated package, for instance, “MySearch” (a-3). The user can use the new package just like a filter. After dragging the package to the edge (b-1), the user obtains the same result as the result without a package (b-2).

3.6 Save the State of Searching

FindFlow provides a function for saving the search status in a file. This supports the user in reusing previous searches.

Sometimes the user starts a search is similar to a previous one. If the search can be continued from the previous one, the user does not have to waste time constructing similar queries.

FindFlow saves the search status automatically. To continue a search, the user has to do is choosing the file. FindFlow reads the file and restores the previous search immediately, and the user can start the new search immediately.

4 FindFlow System

Figure 8 shows the structure of the FindFlow system. FindFlow consists of an interface module and a

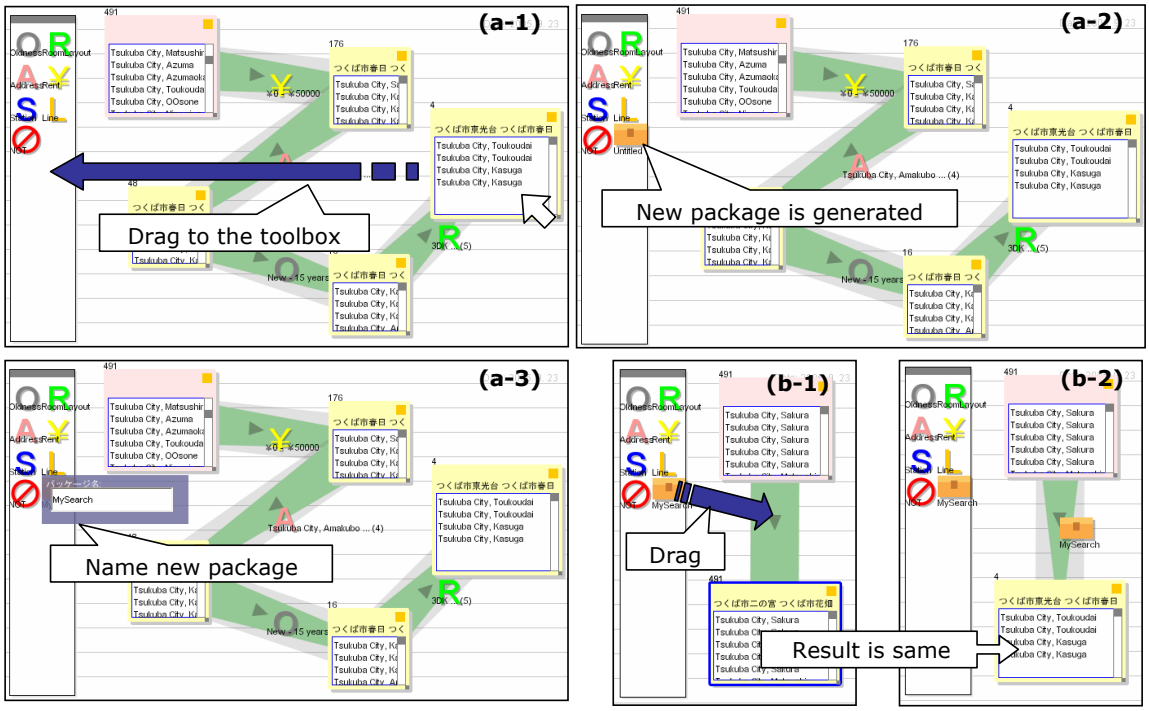

Figure 7: Packaging

database module.

The interface module is operated by the user and it sends queries to the database module. After receiving the result from the database module, it shows it to the user. The interface module comprises the edge, node, and filter. The user manipulates each to construct a query. The database module receives the query from the interface module and converts it into the database query. After conversion, it sends the query to the database and receives the result. Then, it converts the received result into the data for the interface module and returns the converted data as the result.

When the user operates FindFlow, each node requests the query data and weights of filters from the sending nodes. The sending node sends these data, and each edge creates new query data and new weights from the filter and the received data. After receiving the new query data and weights, the node sends the query data to the database module. The database module receives the query data and converts it into a database query. It then sends the converted query to the database server and receives the result. Database module converts the received result for the interface module, and returns the converted result to the node. After receiving the result from the database module, the node sorts the data based on new weights of filters. After that, the interface module shows the result to the user by changing the result in the node, the thickness of edges, and so on.

5 Application Example

5.1 Searching for Apartments in Tsukuba by Trial and Error

We made a search system for rental listings in the city of Tsukuba. We prepared filters for age, layout, address, rent, and station proximity.

Figure 9 shows screenshots of a user searching for apartments in Tsukuba. The criteria are:

- Location is “Amakubo”, “Sakura” or “Kasuga”.
- As new as possible.
- Rent as low as possible.

First, FindFlow shows the screen (a), and the user constructs a query specifying 70,000 yen rent and Amakubo, Sakuraor Kasuga as the location (b). Many results are returned, and the user changes the rent filter, making the rent lower little by little. The user finds that the lowest rent is 30,000 yen (c).

The user focuses attention on the Kasuga area. He wants to push forward the search, but every apartment found is older than 30 years. He therefore stops filtering data (d) and decides to reconsider the whole query.

The user rearranges the query. The user inserts a filter with the condition that apartment should be less than 15 years old (e). He changes the rent filter to 50,000 yen and finds matches with the query (f). He then changes the age condition by requesting a newer apartment (g), and finds the apartment to match the query (h). This listing meets the users requirements.

5.2 Classification of Mails and Files

By regarding the node as a classification folder, FindFlow can serve as a classification tool.

Suppose the user wants to search for or classify some e-mails. FindFlow generates filters automatically from mail headers. The user classifies e-mails just like in searching and saves the search status after finishing the classification. The user loads newly received e-mails to FindFlow. After loading the saved status, FindFlow classifies new e-mails by the query automatically. When the classification does not go well, the user changes the query partly or entirely.

FindFlow is useful for classifying files, too. The user can put files in one place and does not have to classify files with traditional directories. With FindFlow, even when new files are added, the user does not have to be remember where files are placed. FindFlow automatically classifies files by the queries. Because the search status has been saved, the user can load the same status and can find needed files anytime.

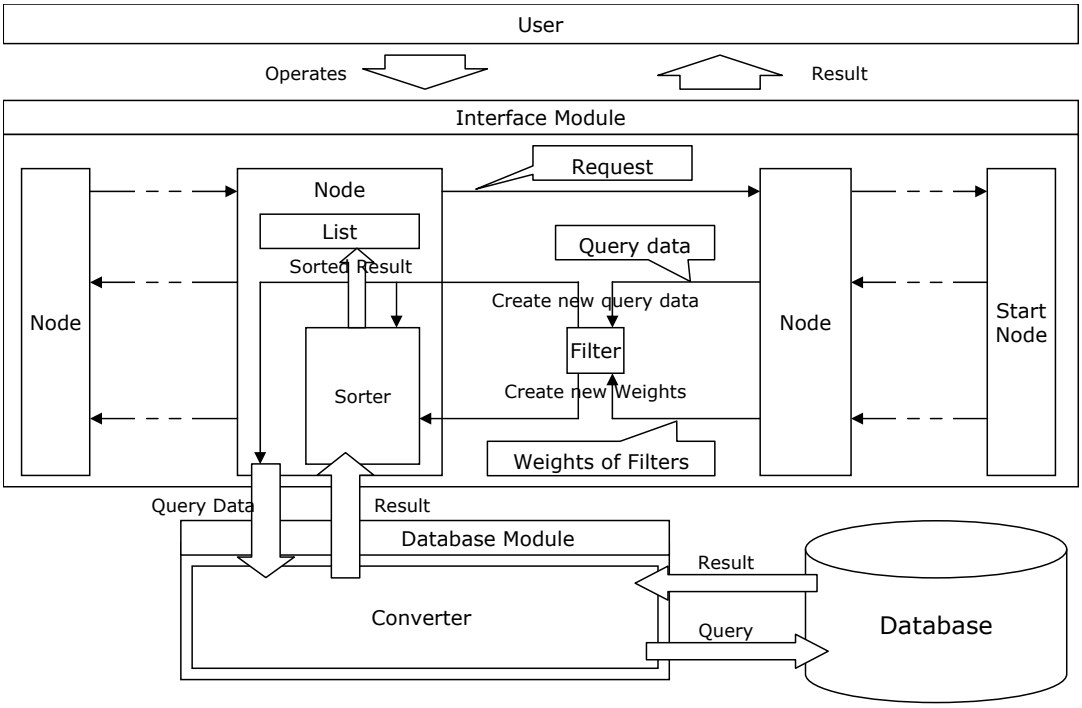

Figure 8: System

6 Discussion

We had various users try FindFlow. The users were businessmen, students, or researchers. After the trials, the users were interviewed. The users found that the operations were simple and easy to understand. They also reported that it was easy to determine which conditions were impeding the search and that it was good that the result was simple. On the negative side, they mentioned that the screen was too small and that, although the node shows the result, it was too small to see and did not contain enough information.

The screen size problem could be solved by providing a scrolling function. Scrolling would allow the user to get more space for query construction. An automatic zooming function would also be useful. When constructed queries reach the edge of screen, FindFlow changes the scale. A zoom would enable users to zoom in on the screen near the mouse pointer.

7 Conclusions and Future Work

We have developed an interactive visual search interface called FindFlow. FindFlow shows not only final results but also intermediate results. The user can change a query based on the intermediate results. FindFlow supports query construction, trial-and-error search, and reuse, enabling the user to construct, recombine and change queries easily, quickly, and properly. FindFlow can be applied to various kinds of searches.

We plan to evaluate FindFlow experimentally to clarify it's advantages and problems. We are also considering how to better use FindFlow in cooperation with the Web and how to improve the methods for showing data.

References

Jaime Teeven, Christine Alvarado, Devid R. Karger & Mark S. Ackerman (2004), The Perfect Search Engine is Not Enough, *in* 'Proceedings of Conference on Human Factors in Computing Systems', pp. 415–421.

Angelaccio, M., Catarci, T., & Santucci, G. (1990), QDB: A Graphical Query Language with Recursion, *in* 'IEEE Transactions on Software Engineering', pp. 1150–1163.

Batini, C., Catarci, T., Costabile, M.F. & Levialdi, S. (1996), Visual Query Systems: A Taxonomy, *in* 'Proceedings of the 2nd IFIP WG2.6 Working Conference on Visual Databases'.

Catarci, T., Costabile, M.F. & Levialdi, S. (1993), On Visual Representations Database Query Systems, *in* 'Proceedings of the Interface to Real and Virtual Worlds Conference', pp. 273–283.

Deepa Krishnan & Tosiyasu L. Kunii (1991), A Visual Query Interface for an Engineering Database, *in* 'Database Systems for Advanced Applications'.

Aiken, A., Chen, J., Lin, M., Spalding, M., Stonebraker, M. & Woodruff, A. (1995), The Tioga-2 Database Visualization Environment, *in* 'Proceedings of the IEEE Visualization Workshop'.

Degi Young & Ben Shneiderman (1993), A Graphical Filter Flow Representation of Boolean Queries: A Prototype Implementation and Evaluation, *in* 'Journal of the American Society for Information Science', Vol. 44(6), pp. 327–339.

Diansheng Guo (2003), A Geographic Visual Query Composer (GVQC) for Accessing Federal Databases, *in* 'Proceedings of National Conference for Digital Government Research', pp. 397–400.

Steven L. Tanimoto (2003), Programming in a Data Factory, *in* 'Proceedings of Human Centric Computing Language and Environments', pp. 100–108.

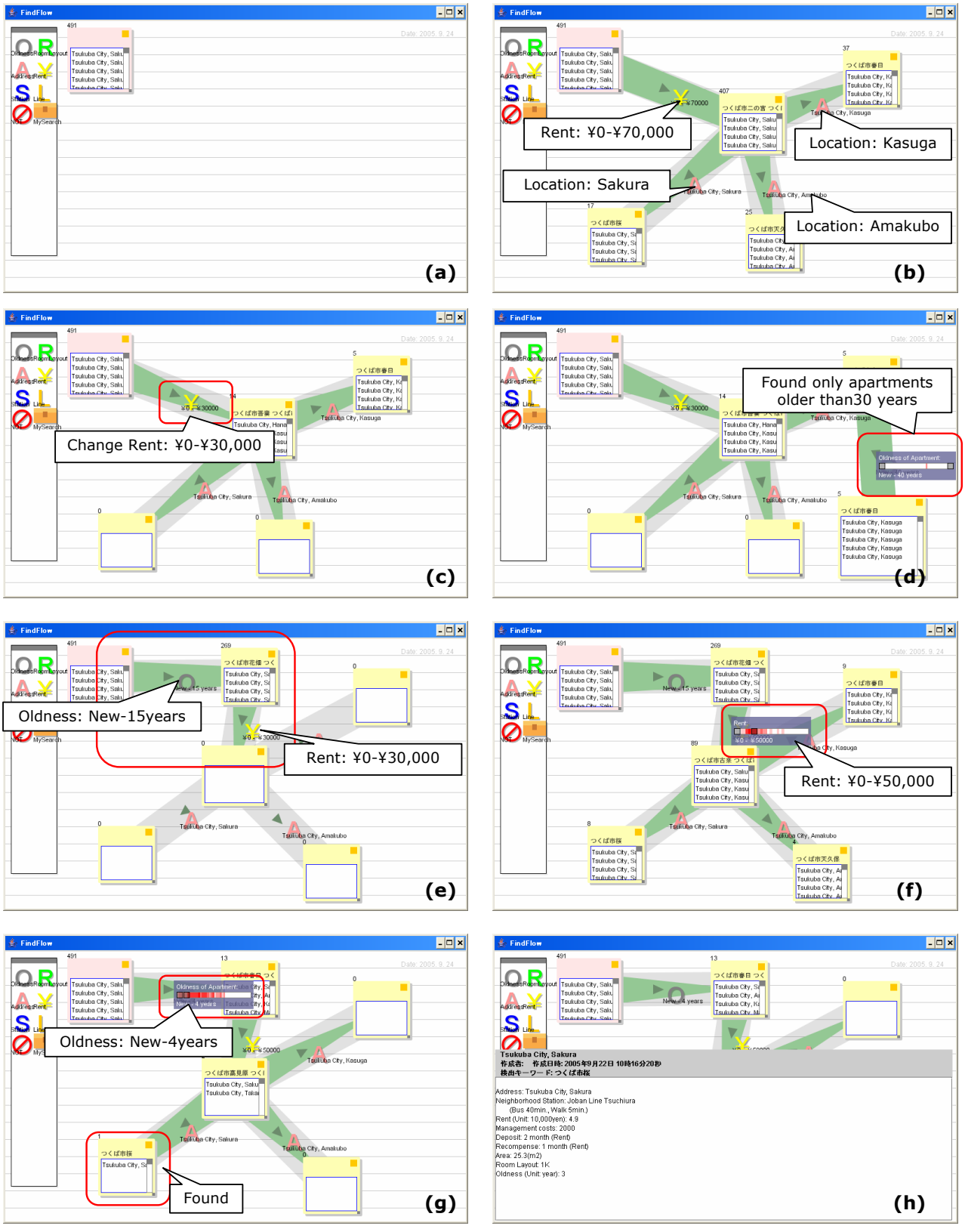

Figure 9: Example of search with FindFlow: Finding an apartment in Tsukuba.