Automatic K-Resources Discovery for Hybrid Web Connected Environments

Comparative Results between DFS and BFS

In this appendix, we present and analyze several tests to compare the performance of both DFS (Depth First Search) and BFS (Breadth First Search) in terms of response time (ms), while searching for a resource providing a specific function. The experiments were conducted on a simulated graph of 2000 resources (i.e., 1000 static and 1000 dynamic), providing functions defined in two different function graphs:

1. Horizontally distributed (**Figure 1**), in which 1000 functions are defined in a directed acyclic function graph distributed horizontally. The function graph consists of 50 branches, each, with 20 ordered functions.
2. Vertically distributed (**Figure 2**), in which 1000 functions are defined in a directed acyclic graph distributed vertically. The function graph consists of 50 dependent functions vertically connected to 2 branches with 450 vertically ordered functions, each.

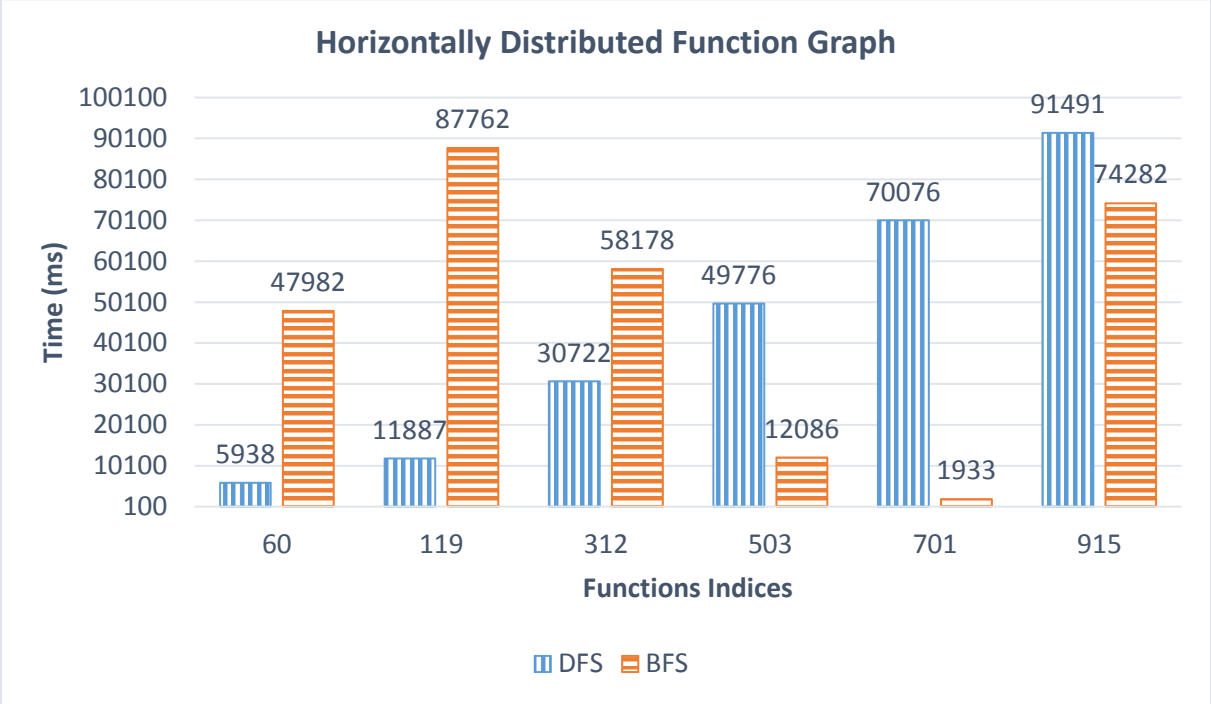

Figure 1- Horizontally distributed function graph

As illustrated in **Figure 1**, when the functions exist in the lower left side of the function graph, such as f_{60}, f_{119}, and f_{312}, DFS is faster than the BFS. This is due to the exploration of the graph resources in a small number of branches of 20 functions, each, with DFS, while in BFS the exploration is done in an important number of levels consisting of 50 functions, each. However, when the functions are in the upper right side of the function graph, as f_{503} and f_{701}, BFS is better than DFS, since it reaches the required function by crawling less number of levels comparing to the number of branches. As for the $f_{915,}$ which is located in the lower right side of the function graph, BFS is still better than DFS, but the difference is not important comparing to the two tests applied for the previous functions (i.e., f_{503} and f_{701}) because there is a big number of branches and levels to cross for both DFS and BFS respectively.

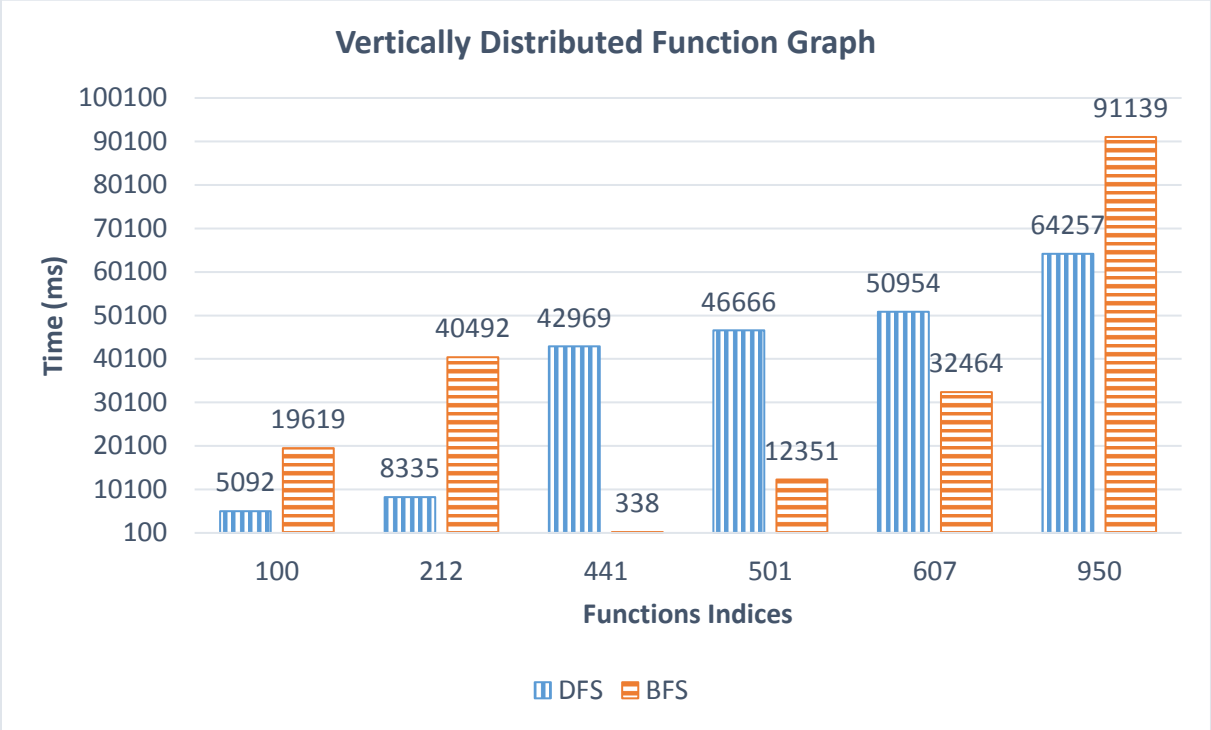

Figure 2- Vertically distributed function graph

In **Figure 2**, DFS is faster than BFS when searching for a resource answering a function that is located in the upper left of the function graph, as f_{100} and f_{212}. The reason is that DFS traverses a single branch and reaches a function with a lower depth comparing to its total big depth, while BFS crawls the graph by levels and thus it moves from a long branch to another before reaching the required function. Nevertheless, BFS is better than DFS when searching for resources providing functions f_{441}, f_{501} and f_{607}. These functions exist in the upper right of the second branch, thus, they can be reached faster when crawling the graph by levels instead of moving deeper in the first branch with DFS. As for f_{950}, DFS is faster than BFS, but still both algorithms time responses are high comparing to the other tests. This is due to the function existence in the lower right side of the function graph, leading to a big number of levels for the BFS, and to an important depth of both branches for the DFS.

The results show that the performance of each algorithm depends on the functions distribution and the localization of the requested function in the function graph, FG. Currently, the algorithm type used to traverse the resource graph is specified by the solution administrator. In a later phase, it will be calculated dynamically based on the function graph topology.