in plain LEDA and GML format is also supported. Furthermore, the system can produce textual listings of the graph and its associated data and either append these listings to a file (useful for logging purposes) or submit them to an arbitrary program for displaying or printing (e.g., `lpr` or `xless`). Graph statistics and tool results can also be logged in delimited format, which is suitable for post-processing with databases, spreadsheets, statistics packages and similar programs. Graphical output of the graph is not (yet) supported, but of course the contents of the main window can be captured and saved or printed using any appropriate X utility such as, e.g., `xgrab`.

Various parameters of the graph and service/requirements generators and the algorithmic tools, as well as display options, can be set according to the user's needs and preferences. These options can be saved in a file named `.dabrc` in the working directory, from which they will be reloaded automatically when the program is reinvoked later.

# 7 Macros, Test Generation and Database Interface

The DABTool program also includes a macro language which makes it possible to implement demos and perform larger test series automatically, and provides logging facilities to obtain a transcript of the computed results, which can then be imported into some spreadsheet or database program. While this scheme provides a high level of flexibility, the programming of test runs and manual postprocessing of the generated data can still be an arduous task. Therefore the DABTool program has some additional facilities for simplifying this process. First, there is a *test configuration dialog* (Figure 2) which handles the generation of common kinds of test macros. With this dialog, the user simply selects the types of problem instances to be generated and the computations to perform; a corresponding test macro is then generated automatically. Second, the DABTool program also provides a *database interface* based on a client/server architecture. Using this interface, all input parameters and the generated test results are exported to a central ADABAS database on a remote server. By these means, different team members can share the same test database, and the test data can also be accessed from a variety of non-UNIX database and spreadsheet applications using the remote SQL interface of the database server.

The test results are stored in the database in a structured, hierarchical format which enables the user to correlate related input parameters and test results in a straightforward manner. The user then accesses the database using external tools such as Microsoft Access and Excel, in order to collect test results, perform statistical analysis and visualization of results using different kinds of diagrams (cf. Figure 3).

# 8 Conclusion and Future Work

The DABTool program has already proved to be a valuable tool for validating our solution techniques. It enables us to test the algorithms with random test instances, from which further insights can be gained on how to improve the solution techniques. Two such random test series are described in [6]. We have also used the program on a realistic problem instance involving 71 services – a hypothetical planning scenario posed by Roland Beutler (Südwestrundfunk, SVP), in which the current FM programmes in South-West Germany were to be "mirrored" in DAB. In this instance, all services had unit bandwidth and had to be fitted into ensembles of size 6. The 71-vertex area graph was extremely dense (2097 edges, density 84.39%, clique