Chapter 4

System Implementation

This chapter describes the system architecture and implementation of the Berth Allocation Planning System (BAPS)

4.1 System Architecture of BAPS

Figure 4.1 shows the architecture of the BAPS. The BAPS comprises 3 subsystems interlinked through a set of data files. The data generation subsystem generates the data files depicting a port, vessel and transshipment scenario; the BAP solver uses these data files as input, solves the BAP and outputs the solution and algorithm trace files; the results analysis subsystem reads the solution and trace files and provides an analysis of the solution quality and algorithm performance. The BAP solver is written in C++ while the data generation and results analysis subsystems

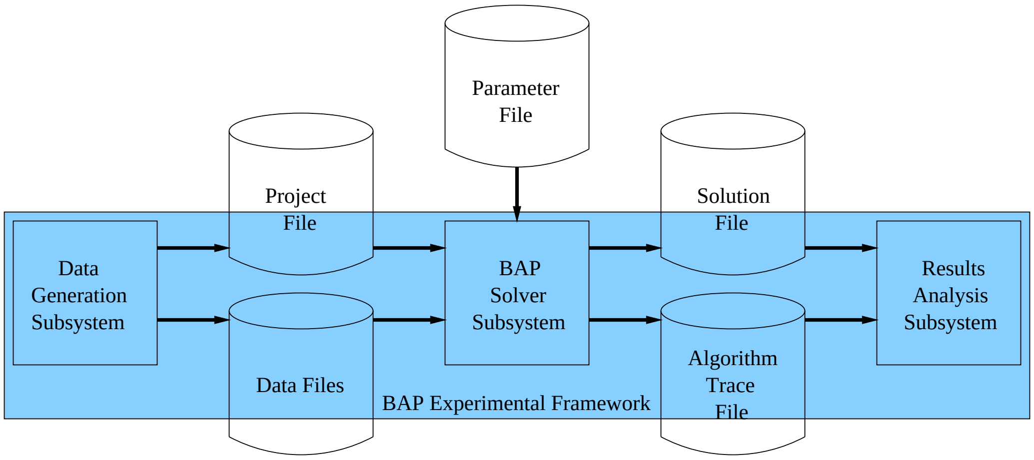

Figure 4.1: BAPS Architecture

are developed using a combination of Unix shell scripts, Perl and C++.

4.2 Data Generation Subsystem

We mentioned in Section 1.2 that real BAP data is classified information and hence we have difficulty obtaining it. However, from the many sources of published information (such as magazines and periodicals), we were able to deduce and reverse-engineer critical information such as the typical port layout, ranges of vessel lengths, average transshipment volumes, vessel schedules, etc ([RAS96]). We then developed a data generation subsystem which allows us to put the above information together and, by specifying a set of parameters, generate different combinations of port, vessel and transshipment scenarios.

Each of these scenarios is captured in a *data set* which is made up of a section file, a vessel file and a transshipment file. The section file describes all the information about the port layout — number of sections, length of each section, the inter-section distances, etc. The vessel file describes the information about a vessel schedule — number of vessels visiting the port, time window of visit, the vessel lengths, arrival, departure times, etc. The transshipment file describes the transshipment information — number of containers to be transshipped from one vessel to another and the number of containers for import/export. For more information on the data generation system, refer to [RAS99].

4.3 BAP Solver Subsystem

As mentioned in Section 1.2, the BAP was modelled to be solved in two stages: the partitioning stage and the packing stage (cf. Figure 1.1). The system we implemented to solve the BAP is known as the *BAPSolver* and its framework is shown in Figure 4.2. In this framework, the *BAPPartitioner* and *BAPPacker* modules implement different algorithms and they are selected during runtime to provide different combinations of algorithms to solve the BAP. The BAPSolver works on *projects*. Each BAP project comprises a data set and a parameter file which describes the parameters which are specific to the algorithm used, such as cross-over value for GA, random-seed for GP, etc. The *project file* captures the specifications (data set, parameter file name, solution file name, etc) of a BAP

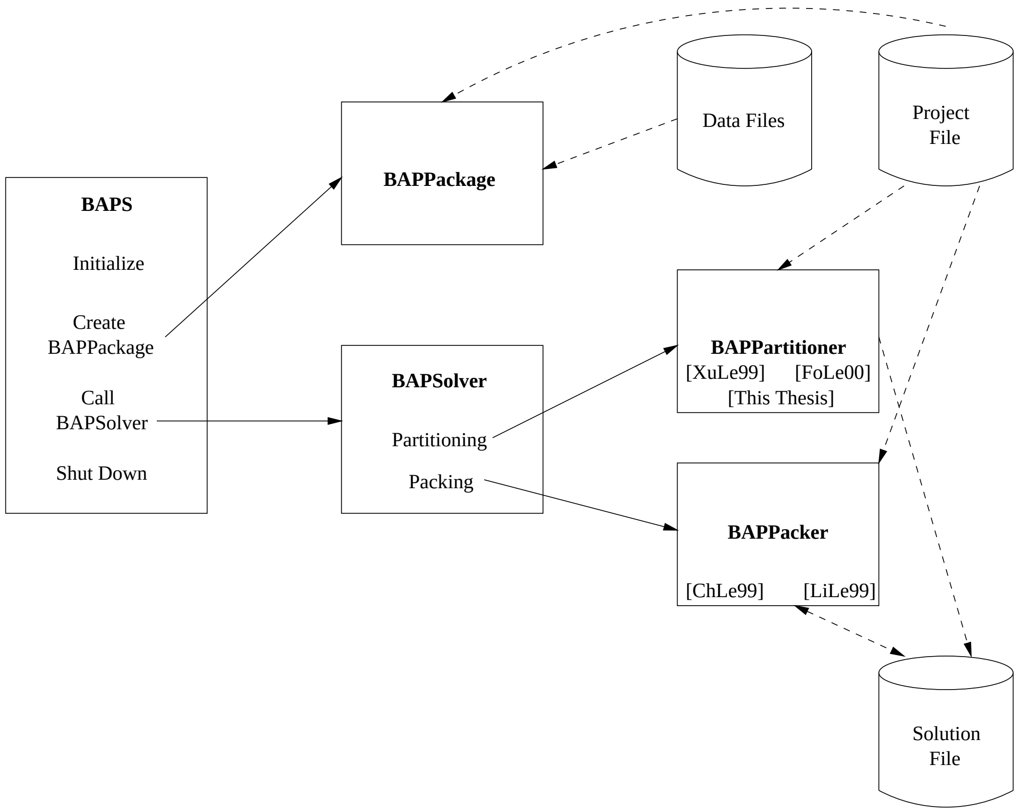

Figure 4.2: BAP Solver Framework

project.

The *BAPPackage* is a runtime object carrying the current project information. After the BAPPackage is created, it is passed to the BAPPartitioner which then solves the current project and updates the BAPPackage with a partial solution with vessels assigned to sections. This partial solution is also output to a partitioning solution file. The BAPPacker then receives the partial solution from the BAPPackage[1] and completes the solution by assigning each vessel to a specific wharf mark within its assigned section. A packing solution file is output as a result. Both the BAPPartitioner and BAPPacker can be configured to output an additional trace file to provide a detailed trace of their algorithms. This information is useful for debugging and for gaining insights into our algorithms. For more detailed information, refer to [RAS99].

4.4 Results Analysis Subsystem

The results analysis subsystem reads the solution files and algorithm trace files and displays an analysis of the solution. In the output, certain information is common across all algorithms (such as the objective values) while certain information is specific to the particular algorithm (such as the number of passes in GP). The

[1] The partial solution can also be read from a partitioning solution file, hence the double arrowhead in the figure.

common information allows us to make comparisons across different algorithms while the algorithm specific information allows us to spot weaknesses and make improvements to our algorithm. The results analysis subsystem provides useful services like automatic tabulation of results to assist us in making comparisons across the different BAP research projects. Refer to [RAS99] for more information.

4.5 Experimental Framework

There is an experimental framework that links the 3 subsystems. The purpose of this framework is to facilitate our experiments by automating the control and information flow across the 3 subsystems. For example, we can generate a data set and automate 100 runs of the solver subsystem on this data set, with each run using a different algorithm parameter, and collect all 100 results for evaluation. With such a framework, we can then run our experiments without supervision.

The experimental framework and the 3 subsystems are described in detail in [RAS99].