Performance Evaluation Lab
Dipartimento di Elettronica e Informazione
Politecnico di Milano - Italy

Java Modelling Tools

users manual

Version 0.4, October 9, 2007

Contents

Chapter 1

Introduction

The Java Modelling Tools (JMT) is a free open source suite for performance evaluation, capacity planning and modelling of computer and communication systems. The suite implements numerous state-of-the-art algorithms for the exact, asymptotic and simulative analysis of queueing network models, either with or without product-form solution. Models can be described either through wizard dialogs or with a graphical user-friendly interface. The suite includes also a workload analysis tool based on clustering techniques. The suite incorporates an XML data layer that enables full reusability of the computational engines.

The JMT suite is composed of *six* tools that support different analyses frequently used in capacity planning studies. The main features of each tool follows.

JSIM: a discrete-event simulator for the analysis of queueing network models. An intuitive sequence of *wizard* windows helps specifying network properties. The simulation engine supports several probability distributions for characterizing service and inter-arrival times. Load-dependent strategies using arbitrary functions of the current queue-length can be specified. JSIM supports state-independent routing strategies, e.g., Markovian or round robin, as well as state-dependent strategies, e.g., routing to the server with minimum utilization, or with the shortest response time, or with minimum queue-length. The simulation engine supports several extended features not allowed in product-form models, namely, finite capacity regions (i.e., blocking), fork-join servers, and priority classes. The analysis of simulation results employs on-line transient detection techniques based on spectral analysis. What-if analyses, where a sequence of simulations is run for different values of parameters, are also possible.

JMODEL: a graphical user-friendly interface for the simulator engine used by JSIM. It integrates the same functionalities of JSIM with an intuitive graphical workspace. This allows an easy description of network structure, as well as a simplified definition of the execution features like blocking regions. Network topologies can be exported in vectorial or raster image formats.

JMVA: meant for the exact analysis of single or multiclass product-form queueing network models, either processing open, closed or mixed workloads. The classic MVA solution algorithm is used. Network structure is specified by textual *wizards*, with conversion functions from probabilities to average visit ratios (and viceversa). What-if analyses are allowed.

JMCH: it applies a simulation technique to solve a single station model, with finite (M/M/1/k) or infinite queue (M/M/1), and shows the underlying Markov Chain. It is also possible to dynamically change the arrival rate and service time of the system.

JABA: a tool for the identification of bottlenecks in closed product-form networks using efficient convex hull algorithms. The tool supports models with up to three job classes. It is possible to identify potential bottlenecks corresponding to the different mixes of customer classes. Models with thousands of queues can be analyzed efficiently. The saturation sectors, i.e., the mixes of customer classes that saturate more than one resource simultaneously, are identified.

JWAT: supports the workload characterization phase, with emphasis on Web log data. Some standard formats for input file are provided (e.g., Apache HTTP log files), and customized formats may also be specified. The imported data can initially be analyzed using descriptive statistical techniques (e.g, means, correlations, pdf histograms, boxplots, scatterplots), either for univariate or multivariate data. Algorithms for data scaling, sample extraction, outlier filtering, k-means and fuzzy k-means clustering for identifying similarities in the input data are provided. These techniques allow to determine cluster centroids, and then estimate mean workload and service demands to be used for model parametrization. The tool includes also an interface to the similarity clustering tool CLUTO.

1.1 Starting the JMT suite

Double click on the JMT icon on your *program group* or *desktop*, or open the *command prompt* and type from the installation directory:

```
java -jar JMT.jar
```

the window of Figure 1.1 will be shown.

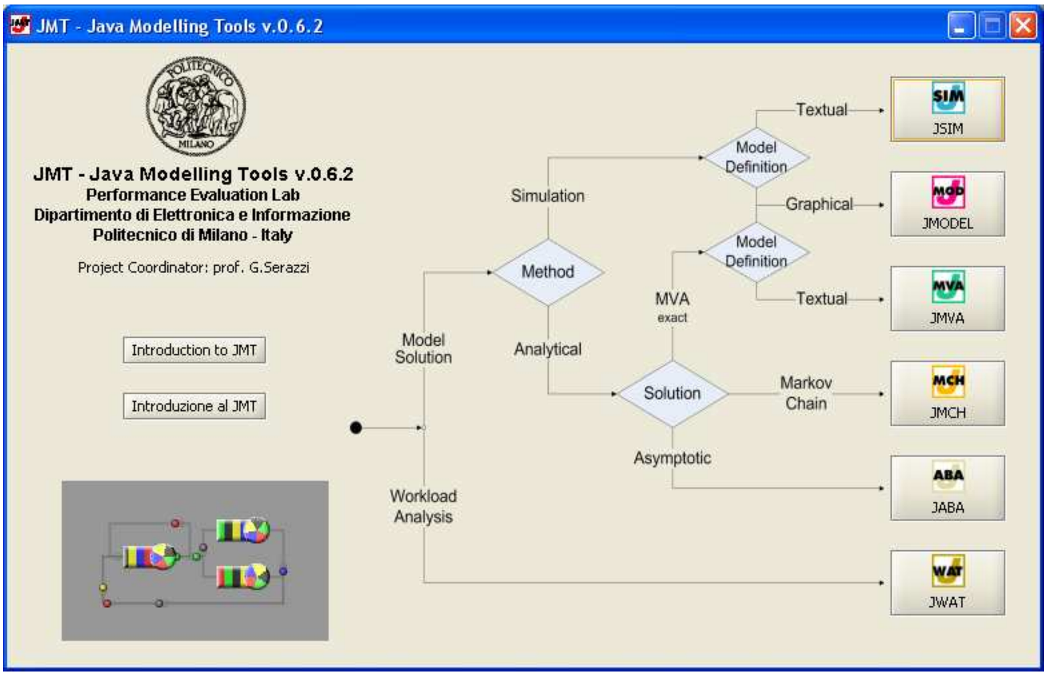


Figure 1.1: JMT suite Starting Screen

This starting screen is used to select the application of the suite to be executed by clicking on the corresponding button. The flow chart try to help the user to chose the application that best fits its needs.

In the following chapters all of the tools[1] will be examined with details and examples. This manual is intended for the general user that wants to learn how to interact with Java Modelling Tools; advanced users that want to learn details on internal data structures, computational engines and XML interfaces should refer to *JMT system manual*.

[1] for the moment only *JMVA*

Chapter 2

JMVA

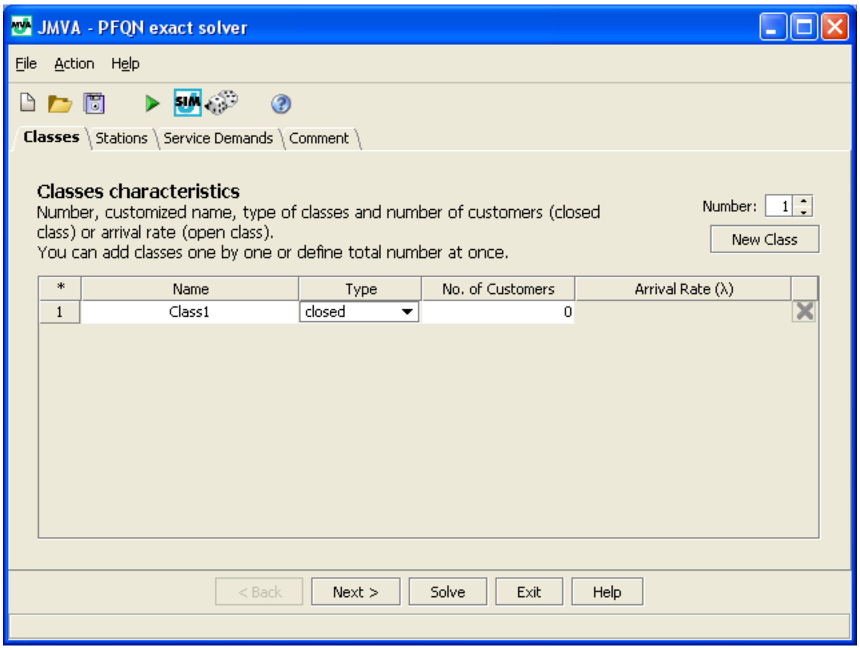
2.1 Overview

JMVA solves open, closed and mixed product form [BCMP75] queueing networks with the exact MVA algorithm [RL80]. In order to avoid fluctuations of the solutions when the model contains load dependent stations, the implemented algorithm is a stabilized version [BBC$^+$81] of the classic MVA algorithm.

Resources may be of two types: *queueing* (either with `load independent` or `load dependent` service times) and `delay`. The model is described in alphanumeric way: user is guided through the definition process by steps of a *wizard* interface (5 or 6 steps). What-if analyses, where a sequence of model are solved for different values of parameters, are also possible (see subsection 2.2.5). A graphical interface to describe the model in a user-friendly environment is also available, see JSIM*graph* for details.

2.1.1 Starting the alphanumeric MVA solver

Selecting MVA button on the starting screen, Figure 2.1 window shows up. Three main areas are shown:



Classes characteristics

Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

*	Name	Type	No. of Customers	Arrival Rate (λ)	
1	Class1	closed	0		✕

Figure 2.1: Classes tab

Menu : it is organized into three groups of functions. To use a menu, click on the menu heading and choose the appropriate option. For the description of menu entries, see section 2.3

Toolbar : contains some buttons to speed up access to JMVA functions (e.g. New model, Open, Save. . . See section 2.3 for details). If you move the mouse pointer over a button a tooltip will be shown up.

Page Area : this is the core of the window. All MVA parameters are grouped in different tabs. You can modify only a subset of them by selecting the right tab, as will be shown later.

2.2 Model definition

Models with one or multiple customer classes provide estimates of performance measures. For a brief description of basic terminology please refer to Appendix A.

In the case of single class models, the workload is characterized by two inputs: the set of service demands, one for each resource, and the workload intensity. On the other hand, in multiple class models, a matrix of service demands is requested [LZGS84].

2.2.1 Defining a new model

To define a new model select toolbar button 🗋 or the *New* command from *File* menu. The following parameters must be defined:

1. `Classes` with their workload intensities (number of customer N for closed classes and arrival rate λ for open classes)
2. `Stations` (service centers)
3. `Service demands` (or Service Times and Visits)
4. Optional short `Comment`

The execution of JMVA provides, for each class and each station, the following performance indices:

- Throughput
- Queue Length
- Residence Time
- Utilization

The following *aggregate* indices are provided:

- System Throughput
- System Response Time
- Average number of customers in the system

Input tabs

As can be seen in Figure 2.1, the parameters that must be entered in order to define a new model are divided in four tabs: `Classes`, `Stations`, `Service Demands` and `Comment`.

Tabs number can become five, if you click *Service Times and Visits* button in `Service Demands Tab`. As will be discussed in subsection 2.2.4, the `Service Demands Tab` will be hidden and it will appear `Service Times Tab` and `Visit Tab`. You can navigate through tabs:

- using sequential wizard buttons, if enabled, at the bottom of the window (Figure 2.2)
- using sequential buttons located in menu
- using the tab selector, clicking on the corresponding tab (Figure 2.3)

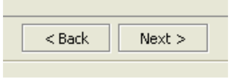


Figure 2.2: Wizard buttons

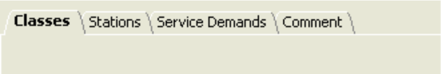
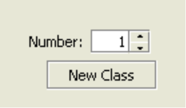


Figure 2.3: Tab selector

2.2.2 Classes Tab

An example screenshot of this tab can be seen in Figure 2.1. This tab allows to characterize customer classes of the model. Your model will be a single class model if and only if there will be only one class, closed or open. On the contrary multiple class models will have at least two classes, closed and/or open.

The number of classes in the model can be specified in the corresponding input area, shown in Figure 2.4 and can be modified using the keyboard or using the spin controls.



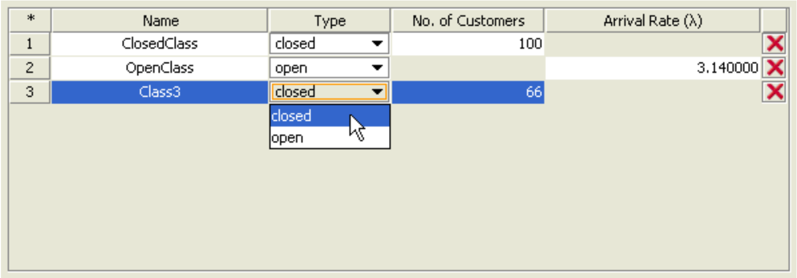
Number: 1

Figure 2.4: Number of classes

Using the delete button ☒ associated to a specific class, a class can be removed provided that there will be at least one class after the deletion. Similar result may be obtained using spin controls, decreasing classes number; in this case last inserted class will be removed.

Default class names are *Class1*, *Class2*, . . . *ClassN*. A model can be personalized by changing this names.

In Figure 2.5 there are three classes of customers, two closed and one open. The third class has the default name *Class3* while the other two classes have personalized names, namely *ClosedClass* and *OpenClass*.

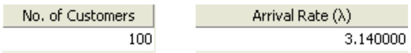


*	Name	Type	No. of Customers	Arrival Rate (λ)	
1	ClosedClass	closed	100		☒
2	OpenClass	open		3.140000	☒
3	Class3	closed	66		☒

Figure 2.5: Defining the classes types

A class type can be `Open` or `Closed`. It is important to define each class type because a closed class workload is described by the number of customers in each class and the open classes workload is described by the customer arrival rate for each class.

As can be seen in Figure 2.5, a class type can be selected in a combo-box. The input boxes *No. of Customers* (N) referring to closed classes accept only positive integer numbers; the input boxes of the *Arrival Rate* (λ) referring to open classes, accept positive real numbers (Figure 2.6).



No. of Customers
100

Arrival Rate (λ)
3.140000

Figure 2.6: Workload definition of the number of customers of a closed class ($N = 100$) and the arrival rate of an open class ($\lambda = 3.14$)

2.2.3 Stations Tab

The number of stations of the model can be specified in the corresponding input area (Figure 2.7) and can be modified using the keyboard or the spin controls.

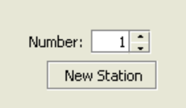
Using the delete button ☒ associated to a specific station, a station can be removed provided that there will be at least one station after the deletion. Similar result may be obtained using spin controls, decreasing stations number; in this case last inserted station will be removed.

Default station names are *Station1*, *Station2*, . . . *StationN*. In order to personalize your model, you can change and give names other than default ones.

In Figure 2.8 there is only one station with default name *Station4* and there are three stations with personalized names: *CPU*, *Disk1* and *Disk2*.

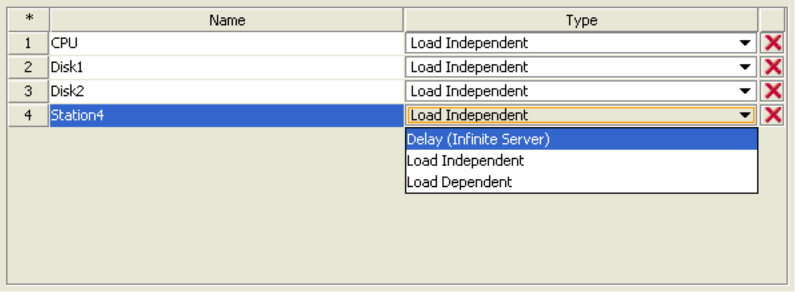
A station type can be `Load Independent`, `Load Dependent` or `Delay`. You can insert in your model a `Load Depend` center only if there is a unique closed class[1]; in all other cases the combo-box will be disabled.

[1] Multiclass, open and mixed models with load dependent stations are not supported yet



Number: 1

Figure 2.7: Number of stations



*	Name	Type
1	CPU	Load Independent
2	Disk1	Load Independent
3	Disk2	Load Independent
4	Station4	Load Independent

Figure 2.8: Defining the stations type

It is important to define each station type because if a station is `Load Dependent` a set of service demand - or a set of service times and the number of visit - must be defined (one service demand/time for each possible value of queue length inside the station).

In subsection 2.2.4 we will explain this concept with more details.

2.2.4 Service Demands, Service Times and Visits Tabs

Service Demands can be defined in two ways:

- directly, by entering Service Demands (D_{kc})
- indirectly, by entering Service Times (S_{kc}) and Visits (V_{kc})

Service demand D_{kc} is the total service requirement, that is the average amount of time that a customer of class c spends in service at station k during one interaction with the system, i.e. it's complete execution. Service time S_{kc} is the average time spent by a customer of class c at station k for a single visit at that station while V_{kc} is the average number of visits at that resource for each interaction with the system.

Remember that $D_{kc} = V_{kc} * S_{kc}$ so it's simple to compute service demands matrix starting from service times and visits matrixes. Inverse calculation is performed with the following algorithm:

$$V_{kj} = \begin{cases} 1 & \text{if} \quad D_{kc} > 0 \\ 0 & \text{if} \quad D_{kc} = 0 \end{cases}$$

$$S_{kc} = \begin{cases} D_{kc} & \text{if} \quad D_{kc} > 0 \\ 0 & \text{if} \quad D_{kc} = 0 \end{cases}$$

Service Demands Tab

In this tab you can insert directly Service Demands D_{kc} for each pair {station k-class c} in the model. In Figure 2.9 a reference screenshot can be seen: notice that a value for every D_{kc} element of the D-matrix has already been specified because default value assigned to newly created stations is zero.

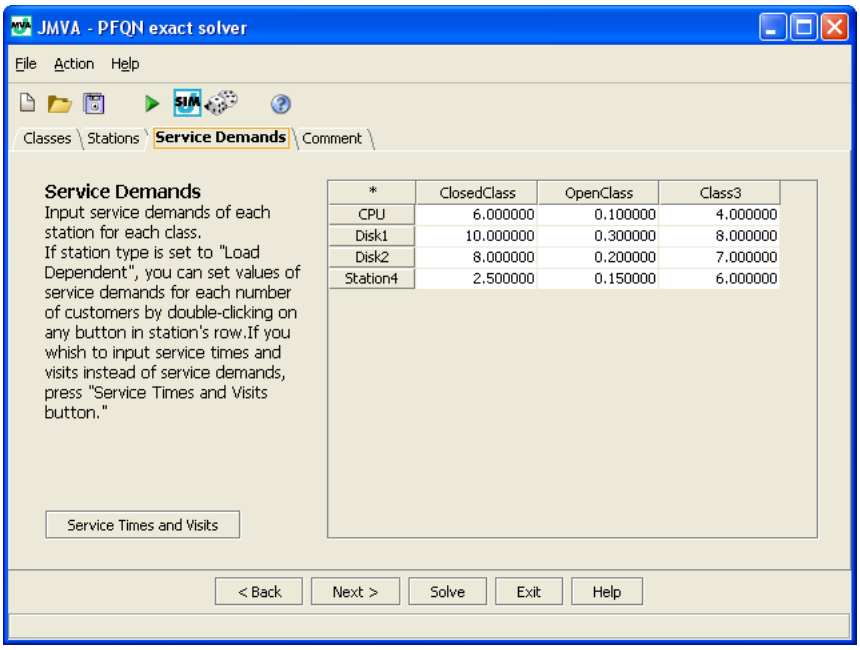
In the example of Figure 2.9, each job of type *ClosedClass* requires an average service demand time of 6 sec to *CPU*, 10 sec to *Disk1*, 8 sec to *Disk2* and 2.5 sec to *Station4*.
On the other hand, a job of type *OpenClass* requires on average 0.1 sec of *CPU* time, 0.3 sec of *Disk1* time, 0.2 sec of *Disk2* time and 0.15 sec of *Station4* time to be processed by the system.

If the model contains any load dependent station, the behavior of Figure 2.10 will be shown.

By double-clicking on *LD Settings...* button a window will show up and that can be used to insert the values of the service demands for each possible number of customer inside the station. That values can be computed by evaluating an analytic function as shown in Figure 2.11. The list of supported operators and more details are reported in subsection 2.2.7.

Service Times and Visits Tabs

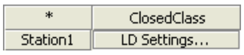
In the former tab you can insert the Service Times S_{kc} for each pair {station k-class c} in the model, in the latter you can enter the visits number V_{kc} (See Figure 2.12).



Service Demands
Input service demands of each station for each class.
If station type is set to "Load Dependent", you can set values of service demands for each number of customers by double-clicking on any button in station's row.If you whish to input service times and visits instead of service demands, press "Service Times and Visits button."

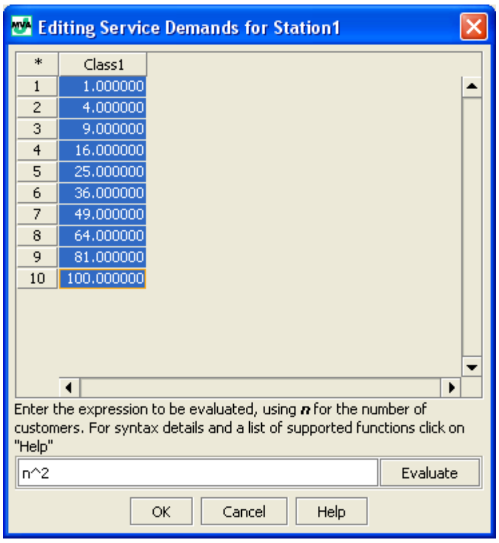
*	ClosedClass	OpenClass	Class3
CPU	6.000000	0.100000	4.000000
Disk1	10.000000	0.300000	8.000000
Disk2	8.000000	0.200000	7.000000
Station4	2.500000	0.150000	6.000000

Figure 2.9: The Service Demands Tab



*	ClosedClass
Station1	LD Settings...

Figure 2.10: Defining a *load dependent* station service demand

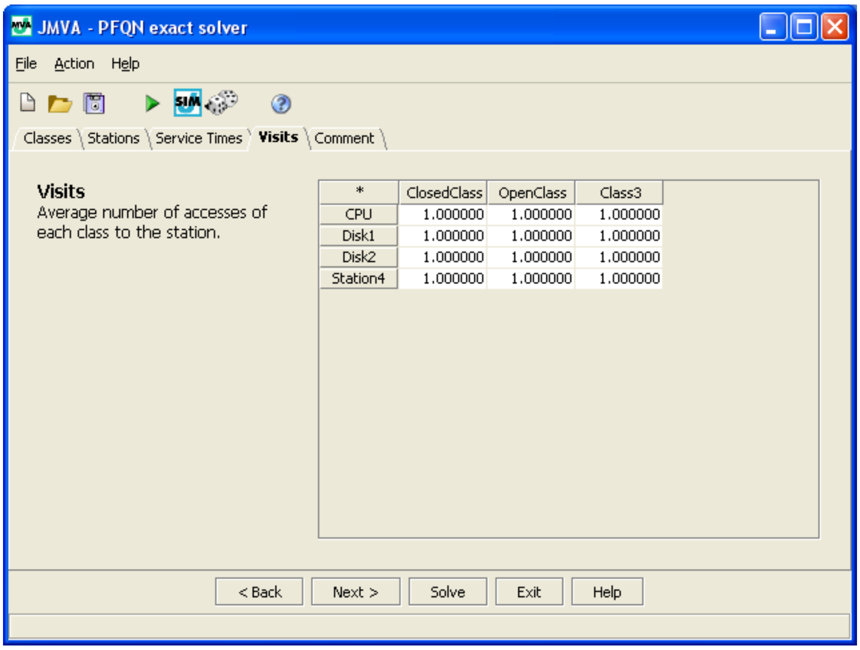


*	Class1
1	1.000000
2	4.000000
3	9.000000
4	16.000000
5	25.000000
6	36.000000
7	49.000000
8	64.000000
9	81.000000
10	100.000000

Enter the expression to be evaluated, using *n* for the number of customers. For syntax details and a list of supported functions click on "Help"

n^2

Figure 2.11: Load Dependent editing window



Visits
Average number of accesses of each class to the station.

*	ClosedClass	OpenClass	Class3
CPU	1.000000	1.000000	1.000000
Disk1	1.000000	1.000000	1.000000
Disk2	1.000000	1.000000	1.000000
Station4	1.000000	1.000000	1.000000

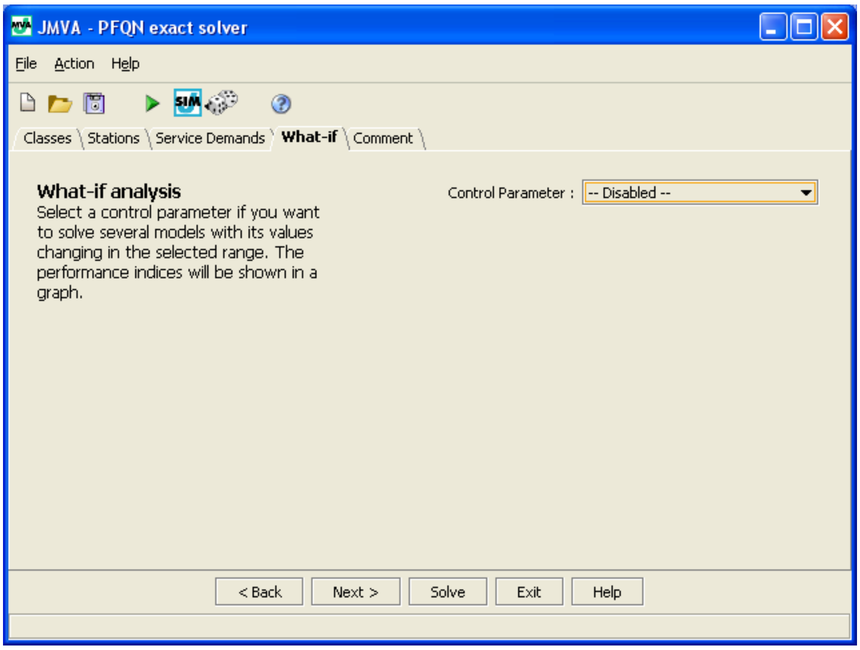
Figure 2.12: Visits Tab

The layout of these tabs is similar to the one of the `Service Demand Tab` described in the previous paragraph. The default value for each element of the Service Times (S) matrix is zero, while it's one for Visits' matrix elements.

In current model contains load dependent stations, the behavior of `Service Times Tab` for their parametrization will be identical to the one described on the previous paragraph for `Service Demands Tab`. On the other hand `Visits Tab` behavior won't change as load dependency is a property of service times and not of visits.

2.2.5 What-if Tab

This Tab is used to perform a what-if analysis, i.e. solve multiple models changing the value of a `control parameter`. In Figure 2.13 is shown this panel when what-if analysis is disabled.



What-if analysis
Select a control parameter if you want to solve several models with its values changing in the selected range. The performance indices will be shown in a graph.

Control Parameter : -- Disabled --

Figure 2.13: What-if Tab - Disabled analysis

The first parameter to be set is the `control parameter` i.e. the parameter that will be changed to solve different models in a selected range. Five choices are possible:

Disabled : disables what-if analysis, so only a single queueing network model, specified in the previous steps, will be solved. This is the *default* option.

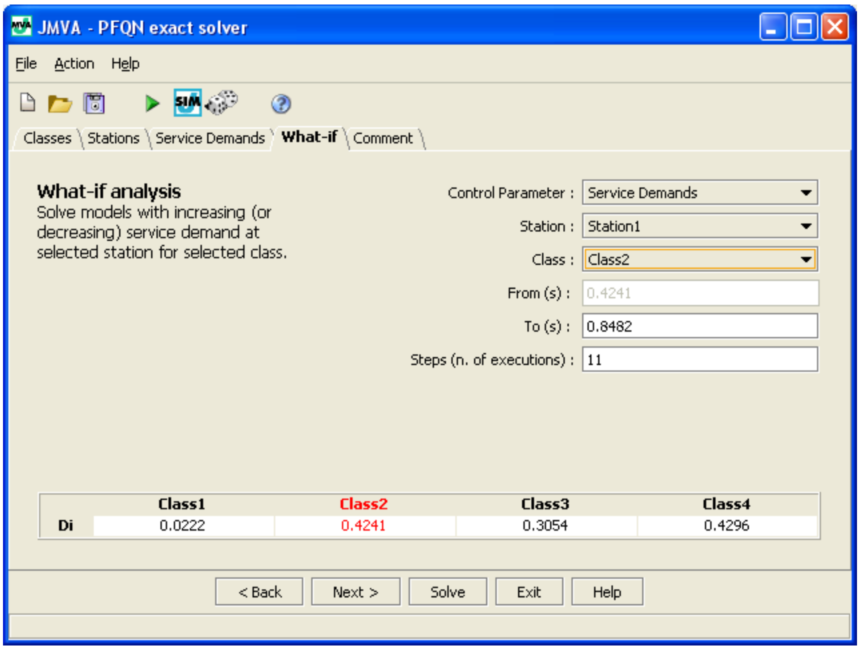
Customer Numbers : different models will be executed by changing the `number of customers` of a single `closed` class or of every closed class proportionally. This option is available only when current model has at least one closed class.

Arrival Rates : different models will be executed by changing `arrival rate` of a single `open` class or of every open class proportionally. This option is available only when current model has at least one open class.

Population Mix : the total number of customers will be kept constant, but the population mix (i.e. the ratio between `number of customers` of selected closed class i and the total number of customers in the system $\beta_i = N_i / \sum_k N_k$). This option is available only when current model has two closed classes.

Service Demands : different models will be solved changing the `service demand` value of a given station for a given class or for all classes proportionally. This option is available only for `load independent` and `delay` stations.

Whenever a control parameter is selected, the window layout will be changed to allow the selection of a valid range of values for it. For example in Figure 2.14 `Service Demands` control parameter was selected. On the bottom of the window, a riepilogative table is presented: depending on selected control parameter, that table is used to show the *initial state* of involved parameters. Every class currently selected for what-if analysis is shown in red.



What-if analysis
Solve models with increasing (or decreasing) service demand at selected station for selected class.

Control Parameter : Service Demands
Station : Station1
Class : Class2
From (s) : 0.4241
To (s) : 0.8482
Steps (n. of executions) : 11

	Class1	Class2	Class3	Class4
Di	0.0222	0.4241	0.3054	0.4296

Figure 2.14: What-if Tab - Service Demands

A brief description of each field is now presented:

Station : available only with `Service Demands` control parameter. This combo box allows to select at which station service demand values will be modified.

Class : allow to select for which class the selected parameter will be changed. A special value, namely `All classes proportionally`, is used to modify the control parameter for each class keeping constant the proportion between different classes[2]. This special value is not available in `Population Mix` analysis as we are changing the proportion of jobs between two closed classes.

From : the initial value of what-if analysis. It was chosen to leave this value fixed to the initial value specified by the user in the previous steps to avoid confusions, so this field acts as a reminder. The only exception is when `Population Mix` is changed, in that case it's allowed to modify this value too.

To : the final value of what-if analysis. Please notice that this value can be greater or smaller than `From` value and is expressed in the same measure unit. Whenever `All classes proportionally` option is selected, both `From` and `To` values are expressed as percentages of initial values (specified in the previous steps

[2] for example, in a model with two closed classes with population vector (2,6), the following models can be executed: (1,3), (2,6), (3,9), (4, 12), . . .

and reminded in the table at the bottom of the panel, see Figure 2.14), in the other situations they are considered as absolute values for the chosen parameter.

Steps : this is chosen number of executions i.e. the number of different models that will be solved. When control parameter is `Customer Numbers` or `Population Mix`, the model can be correctly specified only for integer values of population. JMVA will perform a fast computation to find the maximum allowed number of executions given current `From` and `To` values: if user specify a value bigger than that, JMVA will use the computed value.

2.2.6 Comment Tab

In this Tab, a short - optional - comment about the model can be inserted; it will be saved with the other model parameters.

2.2.7 Expression Evaluator

An expression evaluator is used for the definition of service demands or service times of a load dependent station. It allows to specify times as an analytic function of n where n is the number of customer inside the station.

Expression are evaluated using *JEPLite*[3] (Java Math Expression Parser enlited) package which supports all operators enumerated in Table 2.1 and all functions enumerated in Table 2.2.

Operator	Symbol
Power	^
Unary Plus, Unary Minus	$+n$, $-n$
Modulus	%
Division	/
Multiplication	$*$
Addition, Subtraction	$+$, $-$

Table 2.1: List of all supported operators ordered by priority

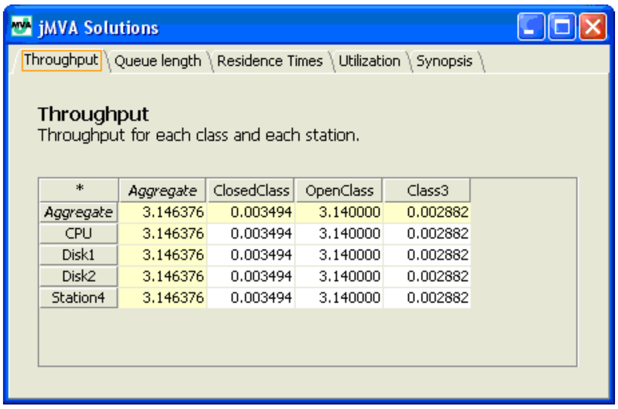
Function	Symbol
Sine	sin()
Cosine	cos()
Tangent	tan()
Arc Sine	asin()
Arc Cosine	acos()
Arc Tangent	atan()
Hyperbolic Sine	sinh()
Hyperbolic Cosine	cosh()
Hyperbolic Tangent	tanh()
Inverse Hyperbolic Sine	asinh()
Inverse Hyperbolic Cosine	acosh()
Inverse Hyperbolic Tangent	atanh()
Natural Logarithm	ln()
Logarithm base 10	log()
Absolute Value / Magnitude	abs()
Random number $[0, 1]$	rand()
Square Root	sqrt()
Sum	sum()

Table 2.2: List of supported functions for the load dependent service times

2.2.8 Model Solution

Use `Solve` command to solve the model. If the model specify a what-if analysis, please refer to subsection 2.2.9. Model results will be shown on a separate window, like the one of Figure 2.15.

[3] http://jeplite.sourceforge.net/



Throughput
Throughput for each class and each station.

*	Aggregate	ClosedClass	OpenClass	Class3
Aggregate	3.146376	0.003494	3.140000	0.002882
CPU	3.146376	0.003494	3.140000	0.002882
Disk1	3.146376	0.003494	3.140000	0.002882
Disk2	3.146376	0.003494	3.140000	0.002882
Station4	3.146376	0.003494	3.140000	0.002882

Figure 2.15: Model Solution (Throughput Tab)

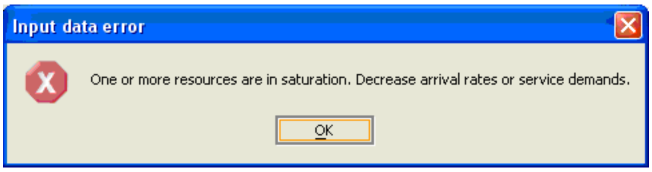
Using the tab selector, all the other computed performance indices can be seen: Throughput, Queue lengths, Residence Times, Utilizations and a synopsis panel with schematic report of input model. Both results and synopsis tab data can be copied to clipboard with `CTRL+C` keyboard shortcut.

When open classes are used, the resource saturation control is performed. For multiple class models, the following inequality must be satisfied:

$$\max_k \sum_c \lambda_c * D_{kc} < 1$$

This inequality ensures that no service center is saturated as a result of the combined loads of all the classes. Let us consider, as example, the model with the classes shown in Figure 2.5 with the D-matrix shown in Figure 2.9 Since $\lambda = 3.14 < 3.33 = 1/0.3 = 1/D_{max}$ the model is not in saturation and the `Solve` command will be executed correctly.

In this example, substituting $D_{\text{Disk1-OpenClass}}$ with values $\geq 1/3.14 \approx 0.318$ will cause the saturation of resource *Disk1* and the error message of Figure 2.16 will appear.



One or more resources are in saturation. Decrease arrival rates or service demands.

Figure 2.16: Input data error message

2.2.9 Model Solution - What-if analysis

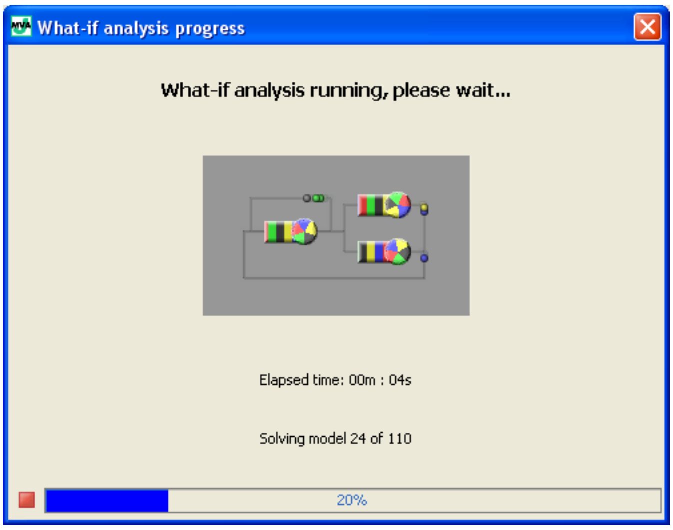
Use `Solve` command to solve the model. During model solution, a progress window, see Figure 2.17, shows up. It displays the cumulative number of models currently solved, the total number of models to be solved and the elapsed time.

At the end of the solution, results will be shown in a separate window, see Figure 2.18. This tab allows to show in a plot the relation between the chosen control parameter (see subsection 2.2.5) and the performance indices computed by the analytic engine.

The combo box `Performance Index` allows to select the performance index to be plotted in the graph, while in the table below, users can select the resource and the class considered.

- The first column is fixed and lists all available colors to be used in the graph.
- The second column, named `Class`, is used to select the class considered in the graph. The special value `Aggregate` is used for the aggregate measure for all classes. If input model is single-class, the class is selected by default for each row.
- The final column, named `Station`, is used to select the station considered in the graph. The special value `Aggregate` is used for the aggregate measure for the entire network. Note that the `Aggregate` value is not valid when the `Utilization` performance index is selected.

In addition to the *center* performance indices (i.e. `Throughput`, `Queue length`, `Residence Times`, `Utilization`), three *system* performance indices are provided in the `Performance Index` combo box (`System Response Time`,



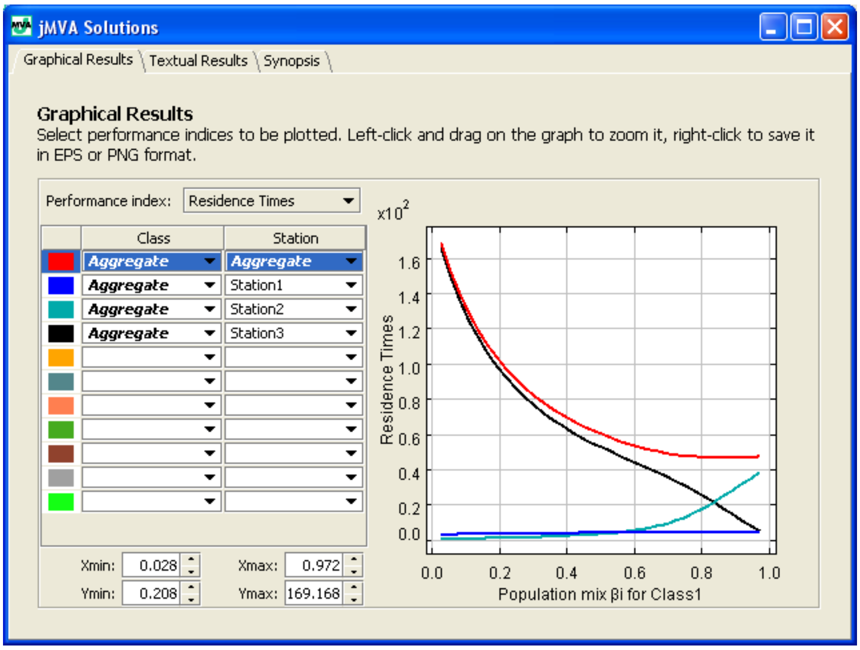
What-if analysis progress

What-if analysis running, please wait...

Elapsed time: 00m : 04s

Solving model 24 of 110

Figure 2.17: Model Solution progress window



jMVA Solutions

Graphical Results

Select performance indices to be plotted. Left-click and drag on the graph to zoom it, right-click to save it in EPS or PNG format.

Performance index: Residence Times

	Class	Station
🟥	Aggregate	Aggregate
🟦	Aggregate	Station1
🟦	Aggregate	Station2
⬛	Aggregate	Station3
🟧		
■		
🟧		
🟩		
🟫		
⬜		
🟩		

Xmin: 0.028 Xmax: 0.972

Ymin: 0.208 Ymax: 169.168

Figure 2.18: Model Solution - Graphical Results Tab

System Throughput, Number of Customers). This *system* indices can be easily obtained by selecting the special Aggregate value for both Class and Station columns of the corresponding center indices (see Appendix A for the definition of the performance indices), but they were provided here as a *shortcut*. As we are referring to aggregate measures, the selection of reference class and station is not significant and, in this case, the table in the left of Figure 2.18 will not be shown.

On the bottom-left corner of the window, users can modify minimum and maximum value of both X and Y axis of the plot. JMVA is designed to automatically best-fit the plot in the window but this controls allow the user to specify a custom range or zoom on the plot. Another fast method to perform a zoom operation is to left-click and drag a rectangle on the graphic window (see Figure 2.19) or right-click on it and select Zoom in or Zoom out options. To automatically reset the best-fit scale users can right-click on the graphic window and select Original view option.

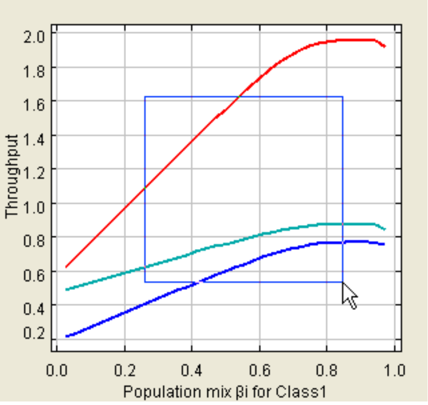
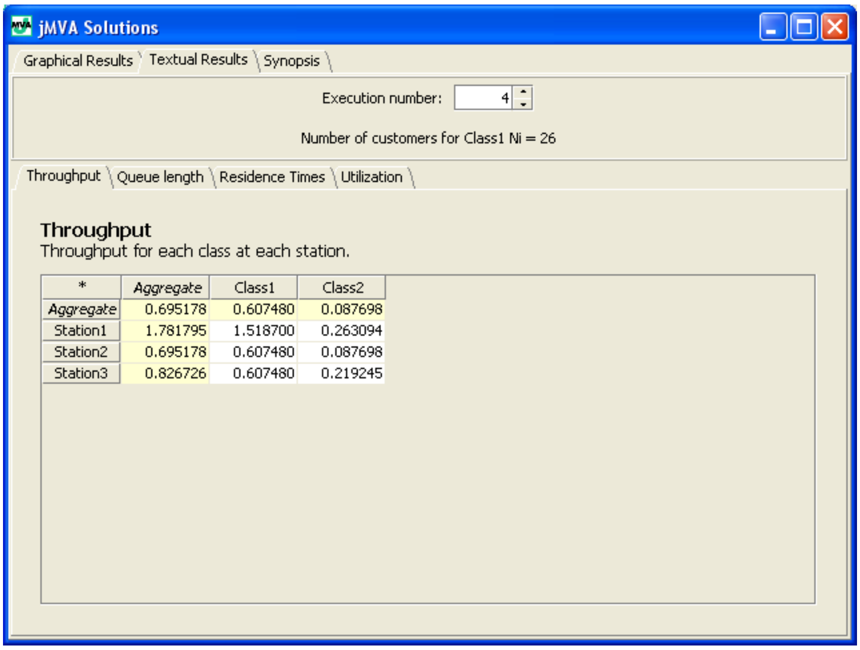


Figure 2.19: Zoom operation on the plot

The graphic window allows to export plots as image to be included in documents and presentations. To save current graph as image, right-click on the graphic and select Save as... option. A dialog will be shown to request the name of the file and the format. Currently supported format are Portable Network Graphics - PNG - (raster) and Encapsulated PostScript - EPS - (vectorial, currently only black and white).

The second tab of the solution window, shown in Figure 2.20, is used to display the solutions of each execution of the analytic algorithm.



Execution number: 4

Number of customers for Class1 Ni = 26

Throughput

Throughput for each class at each station.

*	Aggregate	Class1	Class2
Aggregate	0.695178	0.607480	0.087698
Station1	1.781795	1.518700	0.263094
Station2	0.695178	0.607480	0.087698
Station3	0.826726	0.607480	0.219245

Figure 2.20: Model Solution - Textual Results Tab

This Tab has the same structure of the results window without What-if analysis (described in subsection 2.2.8) but allows to select the execution to be shown in the field Execution Number. By entering requested execution number in the spinner, or using the *up* and *down* arrows, user can cycle between all the computed performance indices for each execution. Just below the spinner, a label gives information on the value of the control parameter for the currently selected execution.

2.2.10 Modification of a model

To modify system parameters return to the main window and enter new data. After the modifications, if you use `Solve` command, a new window with model result will show. You can `save` this new model with the previous name - overwriting the previous one - or `save` it with a different name or in a different directory.

2.3 Menu entries

2.3.1 File

New

Use this command in order to create a new JMVA model.

Shortcut on Toolbar:	🗋
Accelerator Key:	CTRL+N

Open

Use this command to open an existing model. You can only open one model at time, to edit two or more models start more than one instance of JMVA. If current model was modified since its creation or last save action, a popup window will be shown for confirmation.

It's possible to open not only models saved with JMVA (*.jmva), but also with other programs of the suite (for example JABA *.jaba, JSIM *.jsim and JMODEL *.jmodel). Whenever a foreign data file is opened, a conversion is performed and error/warnings occurred during conversion will be reported in a window.

Models are stored in XML format, see *JMT system manual* for a detailed description.

Shortcut on Toolbar:	📂
Accelerator Key:	CTRL+O

Save

Use this command in order to save the active document with its current name in the selected directory.

When you save a document for the first time, JMVA displays the Save As dialog box so you can rename your document. If you save a model after its resolution, results are stored with model definition data.

Shortcut on Toolbar:	💾
Accelerator Key:	CTRL+S

Exit

Use this command in order to end a JMVA session. You can also use the Close command on the application Control menu. If current model was modified since its creation or last save action, a popup window will be shown for confirmation.

Accelerator Key:	CTRL+Q

2.3.2 Action

Solve

Use this command when model description is terminated and you want to start the solution of the model. At the end of the process the window in Figure 2.15 will popup.

Shortcut on Toolbar:	▶
Accelerator Key:	CTRL+L

Randomize

Use this command in order to insert random values into Service Demands - or Service Times - table. Generated values are automatically adjusted to avoid saturation of resources.

Shortcut on Toolbar:	🎲
Accelerator Key:	CTRL+R

Import in JSIM

This command will import current model into JSIM to solve it using simulator. A simple parallel topology is derived from number of visits at each station and generated model is equivalent to original one.

Shortcut on Toolbar:	SIM
Accelerator Key:	CTRL+G

2.3.3 Help

JMVA Help

Use this command to display application help. From the initial window, you can jump to step-by-step instructions that show how use JMVA and consult various types of reference information.

Once you open Help, you can click the Content button whenever you want to return to initial help window.

Shortcut on Toolbar:	?
Accelerator Key:	CTRL+Q

About

Use it in order to display information about JMVA version and credits.

2.4 Examples

In this section we will describe some examples of model parametrization and solution using MVA exact solver. Step-by-step instructions are provided in five examples:

1. A single class closed model with three load independent stations and a delay service center (subsection 2.4.1)
2. A multiclass open model with two classes and three load independent stations (subsection 2.4.2)
3. A single class closed model with a load dependent station and a delay (subsection 2.4.3)
4. A multiclass mixed model with three stations (subsection 2.4.4)
5. A multiclass closed model where a what-if analysis is used to find optimal Population Mix values (subsection 2.4.5)

2.4.1 Example 1 - A model with a single closed class

Solve the single class model specified in Figure 2.21. The customer class, named *ClosedClass* has a population

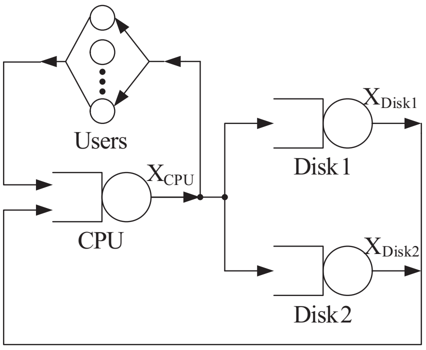


Figure 2.21: Example 1 - network topology

of $N = 3$ customers.

There are four stations, three are of load independent type (named *CPU*, *Disk1* and *Disk2*) and one is of delay type (named *Users*). *Users* delay station represents user's *think time* ($Z = 16$ s) between interaction with the system. Service times and visits for stations are reported in Table 2.3.

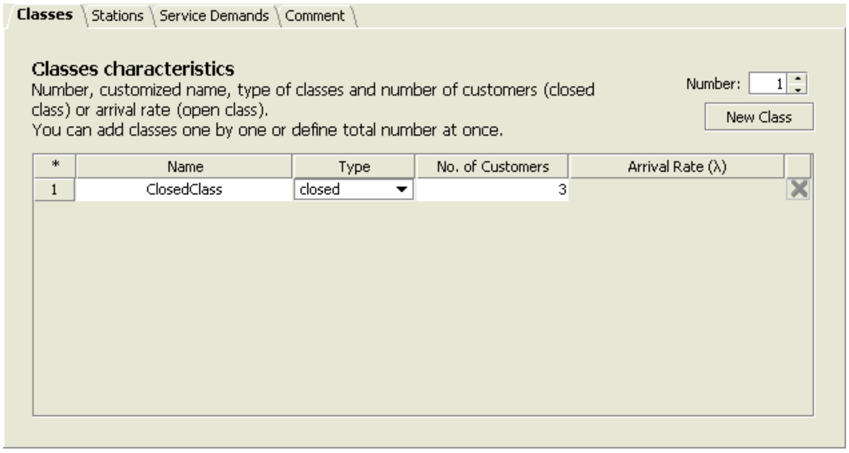
	CPU	Disk1	Disk2	Users
Service Times [s]	0.006	0.038	0.030	16.000
Visits	101.000	60.000	40.000	1.000

Table 2.3: Example 1 - service times and visits

Step 1 - Classes Tab

- use New command to create a new jMVA document
- by default, you have already a `Closed` class
- if you like, substitute default *Class1* name with a customized one (*ClosedClass* in our example)
- complete the table with workload intensity (number of customers). Remember that intensity of a closed class N must be a positive integer number; in this case, 3

At the end of this step, the `Classes Tab` should look like Figure 2.22.



Classes characteristics

Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

*	Name	Type	No. of Customers	Arrival Rate (λ)
1	ClosedClass	closed	3	

Figure 2.22: Example 1 - input data (Classes Tab)

Step 2 - Stations Tab

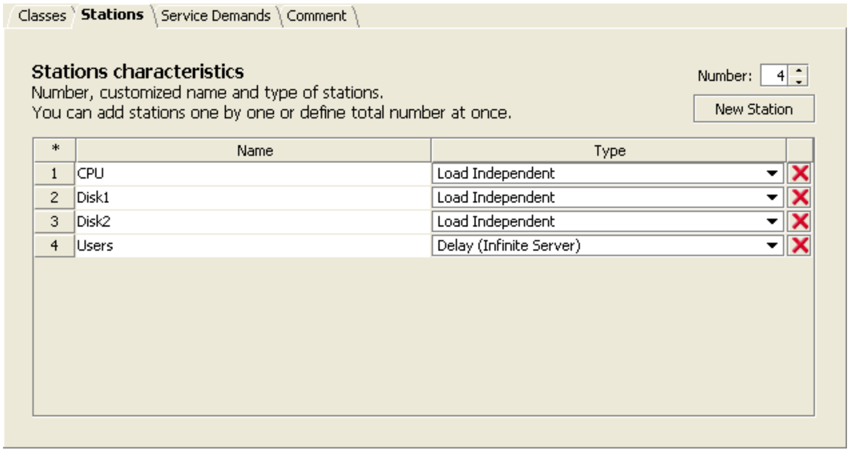
- use `Next` > command to switch to `Stations Tab`
- digit number 4 into stations number textbox or select number 4 using spin controls or push *New Station* button three times. Now your model has four `Load Independent` stations with a default name
- if you want you can change station names. Substitute *CPU* for default name *Station1*, substitute *Disk1* for default name *Station2*, substitute *Disk2* for default name *Station3* and substitute *Users* for default name *Station4*
- change the type of last inserted station; *Users* station is a `Delay (Infinite Server)`

At the end of this step, the `Stations Tab` should look like Figure 2.23.

Step 3 - Service Times and Visits Tabs

- use `Next` > command to switch to `Service Demands Tab`
- press *Service Time and Visit* button as you don't know the Service Demands of the three stations: in this case Service Times and number of Visits should be typed. After button pressure, the `Service Demands Tab` will be hidden and `Service times Tab` and `Visit Tab` will appear
- you can input all Service Times in the table. Remember that Service Time of the *Users* station, of delay type, is *think time Z*, in this case 16 s

At this point, the `Service Times Tab` should look like Figure 2.24.

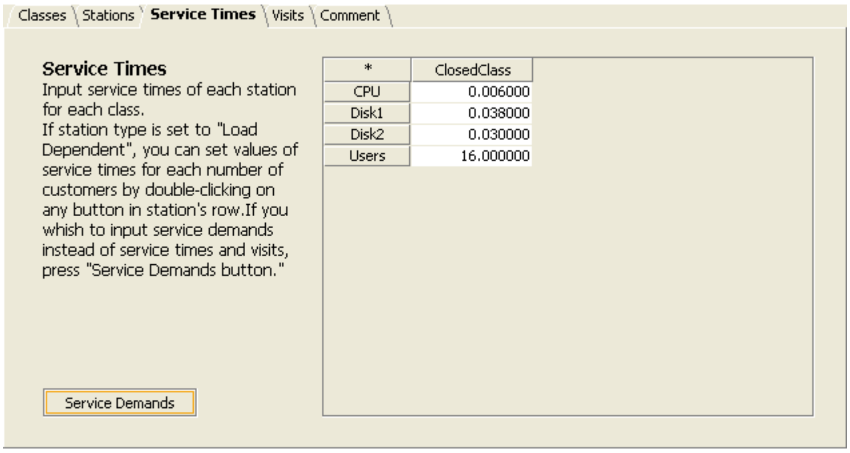


Stations characteristics

Number, customized name and type of stations.
You can add stations one by one or define total number at once.

*	Name	Type
1	CPU	Load Independent
2	Disk1	Load Independent
3	Disk2	Load Independent
4	Users	Delay (Infinite Server)

Figure 2.23: Example 1 - input data (Stations Tab)



Service Times

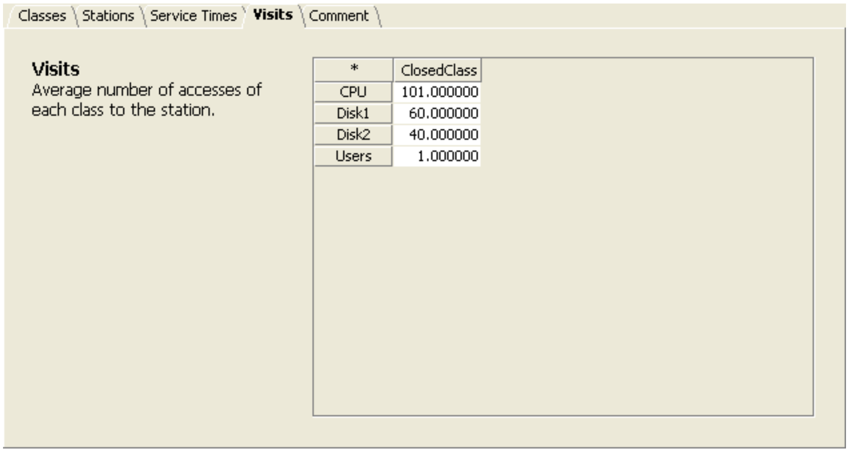
Input service times of each station for each class.
If station type is set to "Load Dependent", you can set values of service times for each number of customers by double-clicking on any button in station's row.If you whish to input service demands instead of service times and visits, press "Service Demands button."

*	ClosedClass
CPU	0.006000
Disk1	0.038000
Disk2	0.030000
Users	16.000000

Figure 2.24: Example 1 - input data (Service Times Tab)

- use `Next >` command to switch to `Visits Tab`
- input numbers of visits for all centers in the table. In this case the number of visits of the *Users*, the infinite server station, is equal to 1 since a customer at the end of an interaction with the system visits this station.

At the end of this step, the `Visits Tab` looks like Figure 2.25.



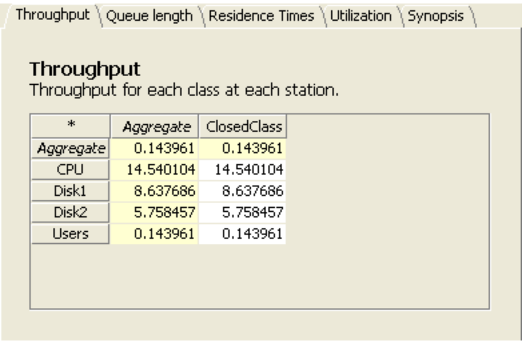
Visits
Average number of accesses of each class to the station.

*	ClosedClass
CPU	101.000000
Disk1	60.000000
Disk2	40.000000
Users	1.000000

Figure 2.25: Example 1 - input data (Visits Tab)

Step 4 - Model Resolution

Use `Solve` command to start the solution of the input model. Model results will be displayed in a new window like the one of Figure 2.26.



Throughput
Throughput for each class at each station.

*	Aggregate	ClosedClass
Aggregate	0.143961	0.143961
CPU	14.540104	14.540104
Disk1	8.637686	8.637686
Disk2	5.758457	5.758457
Users	0.143961	0.143961

Figure 2.26: Example 1 - output data (Throughput Tab)

Since we are considering a single-class model, all results in the column *Aggregate* correspond to the results in the *ClosedClass* column.

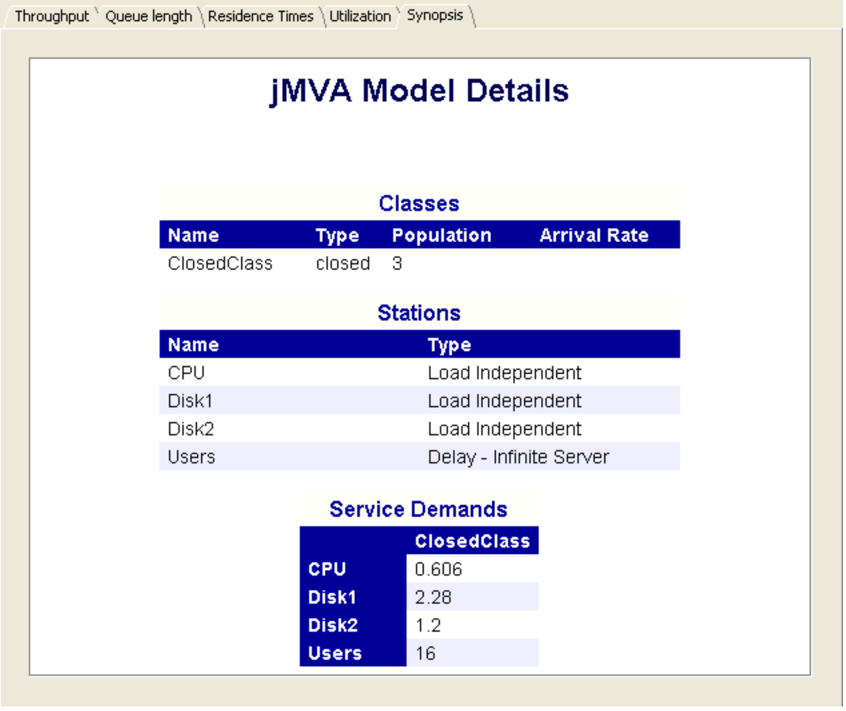
JMVA computes Residence Times W_k, Throughputs X_k, Queue lengths Q_k and Utilizations U_k for all stations. The algorithm begins with the known solution for the network with zero customers, and iterates on N that, in this example, is three. Note that the aggregate *Residence Time* is the *System Response Time* measure and the aggregate *Queue Length* is the average number of customers in the system.

Using tab selector, you can change tab and see Queue length, Residence Times, Utilizations and a synopsis panel with a schematic report of the model (Figure 2.27).

The computed performance indices are shown in Table 2.4.

	Aggregate	CPU	Disk1	Disk2	Users
Throughput [job/s]	0.144	14.540	8.637	5.758	0.144
Queue Length [job]	3.000	0.193	0.410	0.194	2.303
Residence Time [s]	20.839	0.643	2.847	1.349	16.000
Utilization	-	0.087	0.328	0.172	2.303

Table 2.4: Example 1 - model outputs



jMVA Model Details

Classes

Name	Type	Population	Arrival Rate
ClosedClass	closed	3	

Stations

Name	Type
CPU	Load Independent
Disk1	Load Independent
Disk2	Load Independent
Users	Delay - Infinite Server

Service Demands

	ClosedClass
CPU	0.606
Disk1	2.28
Disk2	1.2
Users	16

Figure 2.27: Example 1 - output data (Synopsis Tab)

2.4.2 Example 2 - A model with two open classes

Solve the multiclass open model specified in Figure 2.28. The model is characterized by two open classes A and B with arrival rate (the workload intensity λ) respectively of $\lambda_A = 0.15$ job/s and $\lambda_B = 0.32$ job/s. There are three stations of load independent type, identified with names *CPU*, *Disk1* and *Disk2*. Service times and visits for stations are shown in Table 2.5 and Table 2.6.

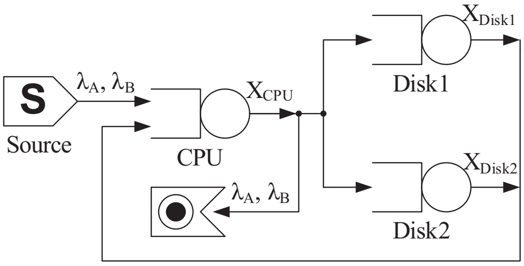


Figure 2.28: Example 2 - network topology

	CPU	Disk1	Disk2
Class A [s]	0.006	0.038	0.030
Class B [s]	0.014	0.062	0.080

Table 2.5: Example 2 - service times

Since this model is similar to the network of Figure 2.21 solved in subsection 2.4.1, we will show how to easily create it from a saved copy of Example 1:

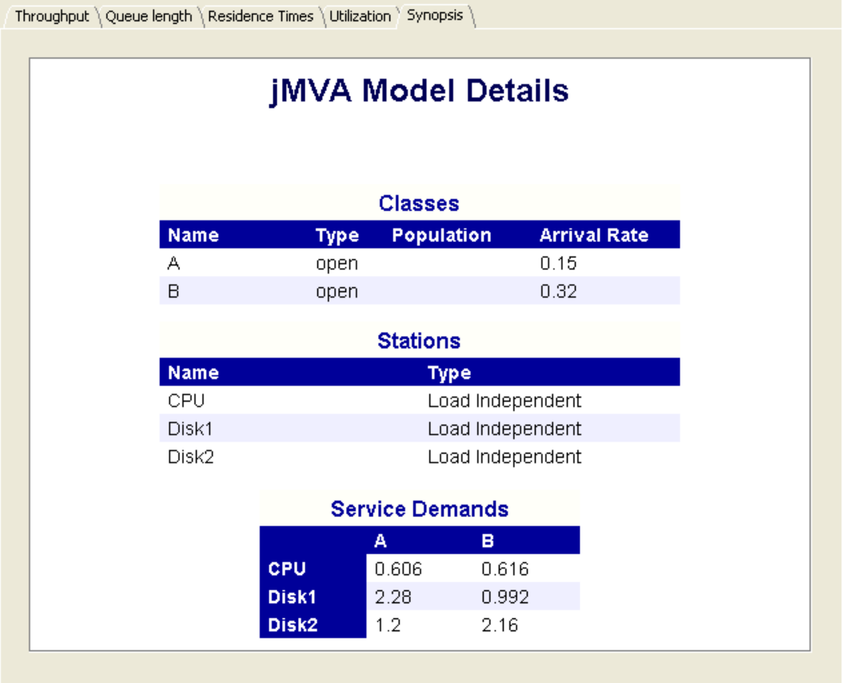
1. Open the saved instance of Example 1 model
2. Go to `Classes Tab`, change *ClosedClass* name to A, change its type to `Open` and set its arrival rate to $\lambda_A = 0.15$ job/s.
3. Click on *New Class* button, sets name of new class to B, change its type to `Open` and set its arrival rate to $\lambda_B = 0.32$ job/s.
4. Go to `Stations Tab` and remove *Users* delay center.
5. Go to `Service Times Tab` and sets service times for Class B according to Table 2.5.
6. Go to `Visits Tab` and sets visits for Class B according to Table 2.6.

	CPU	Disk1	Disk2
Class A	101.0	60.0	40.0
Class B	44.0	16.0	27.0

Table 2.6: Example 2 - number of visits

7. Select `Solve` action.

The `Synopsis Tab` with a schematic report of the model created is shown on Figure 2.29, while the computed performance indices of this model are shown in Table 2.7.



jMVA Model Details

Classes

Name	Type	Population	Arrival Rate
A	open		0.15
B	open		0.32

Stations

Name	Type
CPU	Load Independent
Disk1	Load Independent
Disk2	Load Independent

Service Demands

	A	B
CPU	0.606	0.616
Disk1	2.28	0.992
Disk2	1.2	2.16

Figure 2.29: Example 2 - output data (Synopsis Tab)

		Class A		
	Aggregate	CPU	Disk1	Disk2
Throughput [job/s]	0.150	15.150	9.000	6.000
Queue Length [job]	2.529	0.128	1.004	1.398
Residence Time [s]	16.863	0.851	6.695	9.317
Utilization	-	0.091	0.342	0.180

		Class B		
	Aggregate	CPU	Disk1	Disk2
Throughput [job/s]	0.320	14.080	5.120	8.640
Queue Length [job]	6.575	0.277	0.932	5.366
Residence Time [s]	20.548	0.865	2.913	16.770
Utilization	-	0.197	0.317	0.691

Table 2.7: Example 2 - model outputs

2.4.3 Example 3 - A model with a load dependent station

The network is shown in Figure 2.30. It comprises only two stations: one is of delay type (named *Users*) and the other is a load dependent station (named *Station*). This model has one closed class only with $N = 8$ customers. The user's *think time* is $Z = 21$ s, while the service demands for the load dependent *Station*, shown in Table 2.8, are function of n: number of customers in the station ($D(n) = n + 1/n$).

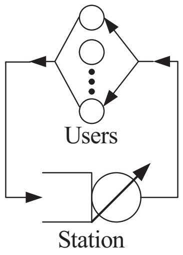


Figure 2.30: Example 3 - network topology

n	1	2	3	4	5	6	7	8
$D(n)$ [s]	2.00	2.50	3.33	4.25	5.20	6.17	7.14	8.13

Table 2.8: Example 3 - service demands for *Station*, a load-dependent service center

Step 1 - Classes Tab

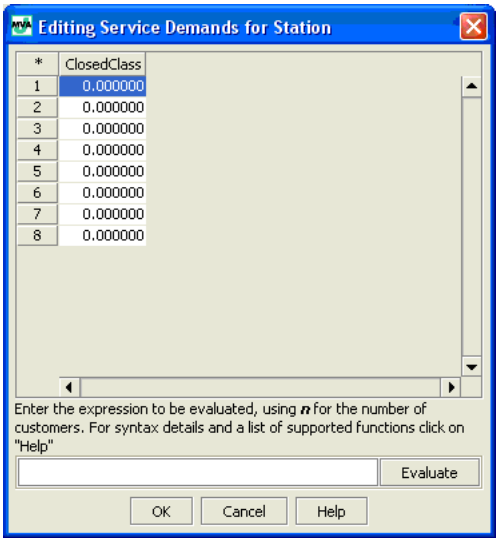
The instructions that are the same given in Step 1 of subsection 2.4.1; in this case N must be 8.

Step 2 - Stations Tab

The instructions are given in Step 2 of subsection 2.4.1; in this case the model has two stations: a Load Dependent station and a Delay Center.

Step 3 - Service Demands Tab

1. use `Next >` command to switch to `Service Demands Tab`
2. double-click on cell with text *LD Setting...* to open the editor of load dependent service demands in a separate window, shown in Figure 2.31.
3. it is not mandatory to insert all values one-by-one. You can click or drag to select cells, enter the expression $n + 1/n$ into the textbox at the bottom of the window and click the *Evaluate* button.
4. at the end of this phase, editor window looks like Figure 2.32. Now you may press *OK* button to confirm changes and return to JMVA main window.



Editing Service Demands for Station

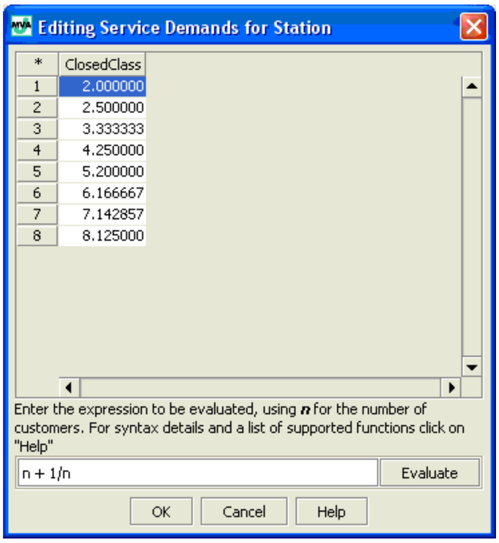
*	ClosedClass
1	0.000000
2	0.000000
3	0.000000
4	0.000000
5	0.000000
6	0.000000
7	0.000000
8	0.000000

Enter the expression to be evaluated, using *n* for the number of customers. For syntax details and a list of supported functions click on "Help"

Figure 2.31: Example 3 - editor for the description of service demands for a load dependent station corresponding to the different number of customers before the parametrization

Step 4 - Model Resolution

Use `Solve` command to resolve the model, results are shown in Table 2.9.



*	ClosedClass
1	2.000000
2	2.500000
3	3.333333
4	4.250000
5	5.200000
6	6.166667
7	7.142857
8	8.125000

Enter the expression to be evaluated, using ***n*** for the number of customers. For syntax details and a list of supported functions click on "Help"

n + 1/n

Figure 2.32: Example 3 - editor for the description of service demands for a load dependent station. In this case an arithmetic function has been defined: $S(n) = n + 1/n$

	Aggregate	Station	Users
Throughput [job/s]	0.234	0.234	0.234
Queue Length [job]	8.000	3.080	4.920
Residence Time [s]	34.149	13.149	21.000
Utilization	-	0.810	0.973

Table 2.9: Example 4 - model outputs

2.4.4 Example 4 - A model with one open and one closed class

The mixed queueing network model is shown in Figure 2.33. Workload intensities: the open class has an arrival rate $\lambda = 1$ job/s, the closed class has a customers number $N = 57$. Service demands are shown in Table 2.10.

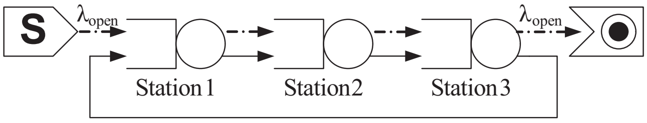


Figure 2.33: Example 4 - network topology

Step 1 - Classes Tab

Follow the instructions of Step 1 in the previous examples; the `Classes Tab` is shown in Figure 2.34.

Step 2 - Stations Tab

Follow the instructions of Step 2 in the previous examples; in this case the model has three Load Independent stations (see Figure 2.35).

Step 3 - Service Demands Tab

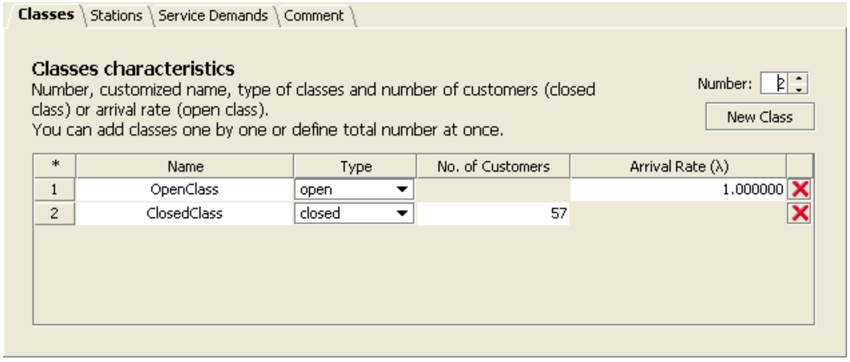
Follow the instructions of Step 3 in the previous examples and define service demands for both classes as illustrated in Table 2.10 (see Figure 2.36).

Step 4 - Model Solution

Use `Solve` command. Results can be verified by computing the *equivalent model*, where the open class "slows down" the closed class by subtracting utilization to it:

	Station1	Station2	Station3
OpenClass [s]	0.5	0.8	0.6
ClosedClass [s]	10.0	4.0	8.0

Table 2.10: Example 4 - service demands



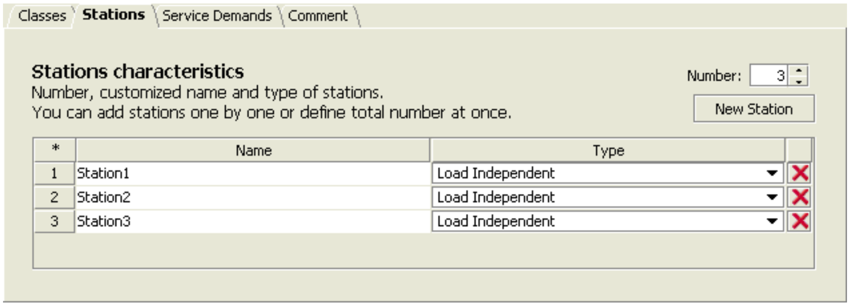
Classes characteristics

Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

Number: 2

*	Name	Type	No. of Customers	Arrival Rate (λ)	
1	OpenClass	open		1.000000	☒
2	ClosedClass	closed	57		☒

Figure 2.34: Example 4 - Class Tab



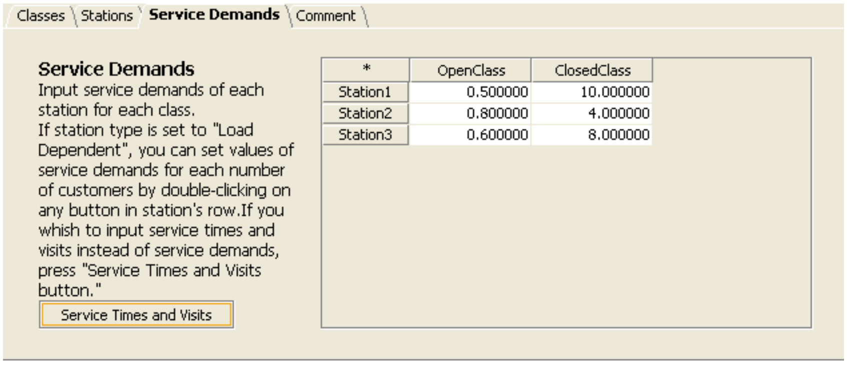
Stations characteristics

Number, customized name and type of stations.
You can add stations one by one or define total number at once.

Number: 3

*	Name	Type	
1	Station1	Load Independent	☒
2	Station2	Load Independent	☒
3	Station3	Load Independent	☒

Figure 2.35: Example 4 - Stations Tab



Service Demands

Input service demands of each station for each class.
If station type is set to "Load Dependent", you can set values of service demands for each number of customers by double-clicking on any button in station's row.If you whish to input service times and visits instead of service demands, press "Service Times and Visits button."

*	OpenClass	ClosedClass
Station1	0.500000	10.000000
Station2	0.800000	4.000000
Station3	0.600000	8.000000

Figure 2.36: Example 4 - Service Demands Tab

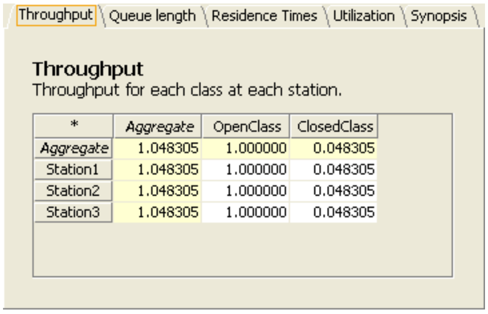
$$\begin{aligned}
D_1^{eq} &= \frac{D_{1,ClosedClass}}{1-\lambda * D_{1,OpenClass}} = 20s \\
D_2^{eq} &= \frac{D_{2,ClosedClass}}{1-\lambda * D_{2,OpenClass}} = 20s \\
D_3^{eq} &= \frac{D_{3,ClosedClass}}{1-\lambda * D_{3,OpenClass}} = 20s
\end{aligned}$$

MVA algorithm is used to solve the equivalent closed model. The number of customers of the closed class is 57, and the exact MVA technique should require the solution of other 56 models with smaller population. In this particular case, the formula used to compute the throughput can be simplified because the Service Demands are all equals:

$$\begin{aligned}
X^{eq}(N) &= \frac{N}{\sum_{k=1}^{3} D_k^{eq} + \sum_{k=1}^{3} [D_k^{eq} * Q_k^{eq}(N-1)]} \\
&= \frac{N}{60 + 20 * \sum_{k=1}^{3} Q_k^{eq}(N-1)} \\
&= \frac{N}{60 + 20 * (N-1)}
\end{aligned}$$

$$X^{eq}(57) = 0.048305 \text{ job/s}$$

So the throughput measure for the closed class is $X_{ClosedClass} = X^{eq} = 0.048305$ job/s while the throughput for the open class coincide with its arrival rate $X_{OpenClass} = \lambda$. As visits were not specified, they have been considered equal to one: that's why throughput is equal at each station for each class (see Figure 2.37), i.e. the solved model consists of 3 stations that are sequentially connected with feedback.



Throughput
Throughput for each class at each station.

*	Aggregate	OpenClass	ClosedClass
Aggregate	1.048305	1.000000	0.048305
Station1	1.048305	1.000000	0.048305
Station2	1.048305	1.000000	0.048305
Station3	1.048305	1.000000	0.048305

Figure 2.37: Example 4 - throughput

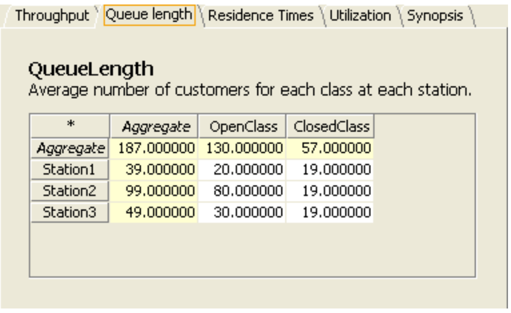
Queue lenghts can be computed with the following formulas:

$$\begin{aligned}
Q_{k,ClosedClass}(N) &= Q_k^{eq}(N) \\
Q_{k,OpenClass}(N) &= \frac{\lambda * D_{k,OpenClass} * [1 + Q_{k,ClosedClass}(N)]}{1-\lambda * D_{k,OpenClass}}
\end{aligned}$$

And the results will be equal to the ones shown in Figure 2.38.

2.4.5 Example 5 - Find optimal Population Mix values

Perform a what-if analysis to find the value of population mix that will maximize `System Throughput` and minimize `System Response Time` in the model shown in Figure 2.39. This model has two closed classes (named *Class1* and *Class2*) with a total population of $N = 20$ and three load independent stations (named *Station1*, *Station2* and *Station3*) with the service demands shown in Table 2.11.



QueueLength

Average number of customers for each class at each station.

*	Aggregate	OpenClass	ClosedClass
Aggregate	187.000000	130.000000	57.000000
Station1	39.000000	20.000000	19.000000
Station2	99.000000	80.000000	19.000000
Station3	49.000000	30.000000	19.000000

Figure 2.38: Example 4 - queue lengths

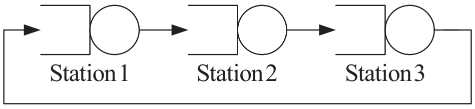


Figure 2.39: Example 5 - network topology

Step 1 - Classes Tab

Follow the instructions of Step 1 in the previous examples; as we will change population mix, initial allocation of the $N = 20$ jobs is irrelevant. For example we can allocate $N_1 = 10$ jobs to *Class1* and $N_2 = 10$ jobs to *Class2*. The `Classes Tab` is shown in Figure 2.40.

Step 2 - Stations Tab

Follow the instructions of Step 2 in the previous examples; in this case the model has three Load Independent stations (see Figure 2.41).

Step 3 - Service Demands Tab

Follow the instructions of Step 3 in the previous examples and define service demands for both classes as illustrated in Table 2.11 (see Figure 2.42).

Step 4 - What-if Tab

1. use `Next` > command to switch to `What-if Tab`
2. Select `Population Mix` as a *control parameter* of the analysis in the combo box, several fields will be shown below.
3. *Class1* is already selected, by default, as reference class for the what-if analysis. This means that β_i values in `From` and `To` fields are referred to *Class1*.
4. By default, JMVA suggests the minimum allowed value of β_1 in the `From` field and its the maximum value in the `To` field[4]. Since we want to find the optimal value in the entire interval, we leave this unchanged.
5. we want to perform the maximum number of allowed executions, so we enter a big number in the `Steps` field (100 for example). JMVA will automatically calculate the maximum number of allowed executions provided that number of customers for each class must be an integer and will report 19.

At the end of this phase, the What-if Tab will look like Figure 2.43.

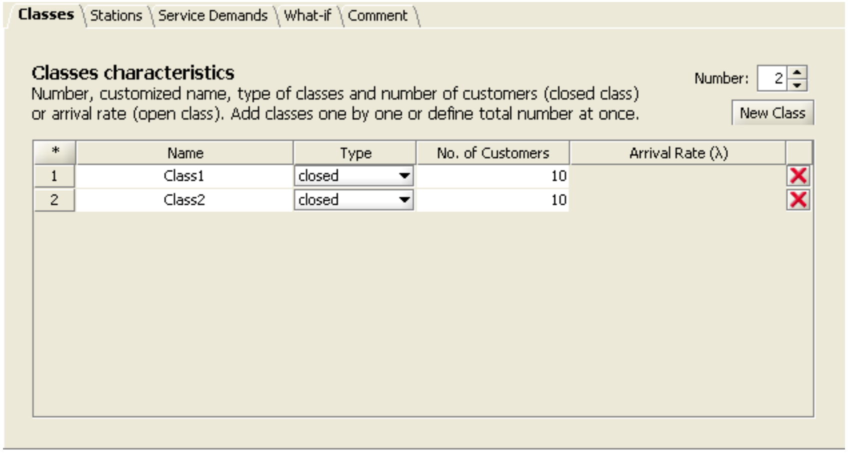
Step 5 - Model Solution

Use `Solve` command. In the `Graphical Results Tab` select `System Response Time` and `System Throughput` as *Performance Index* (Figure 2.44 and Figure 2.45). Zooming on the plot, allows to identify the maximum

[4] Since it is requested that one class has at least one job and customer number must be integer, the minimum value is $1/N$ and the maximum value is $(N-1)/N$.

	Station1	Station2	Station3
Class1 [s]	1.0	5.0	1.0
Class2 [s]	5.0	1.0	5.0

Table 2.11: Example 5 - service demands



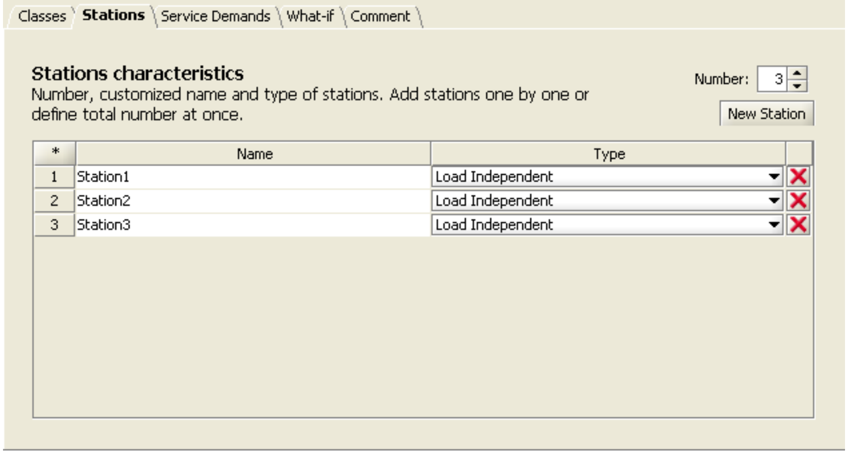
Classes characteristics

Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class). Add classes one by one or define total number at once.

Number: 2

*	Name	Type	No. of Customers	Arrival Rate (λ)	
1	Class1	closed	10		☒
2	Class2	closed	10		☒

Figure 2.40: Example 5 - Class Tab



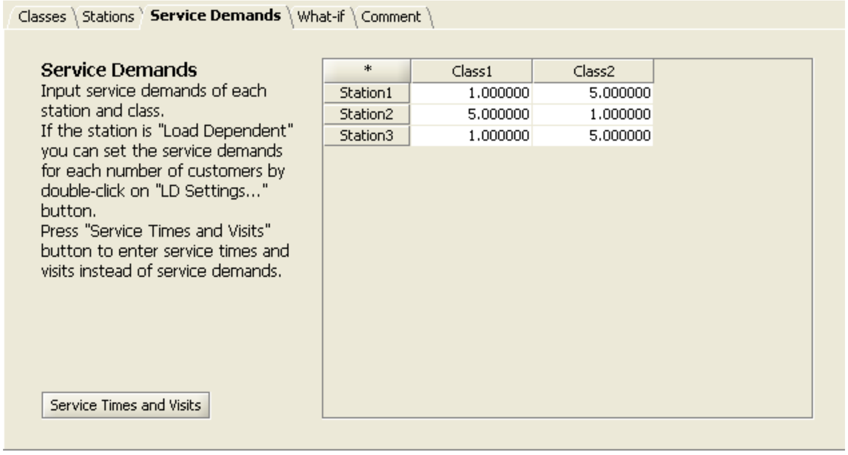
Stations characteristics

Number, customized name and type of stations. Add stations one by one or define total number at once.

Number: 3

*	Name	Type	
1	Station1	Load Independent	☒
2	Station2	Load Independent	☒
3	Station3	Load Independent	☒

Figure 2.41: Example 5 - Stations Tab

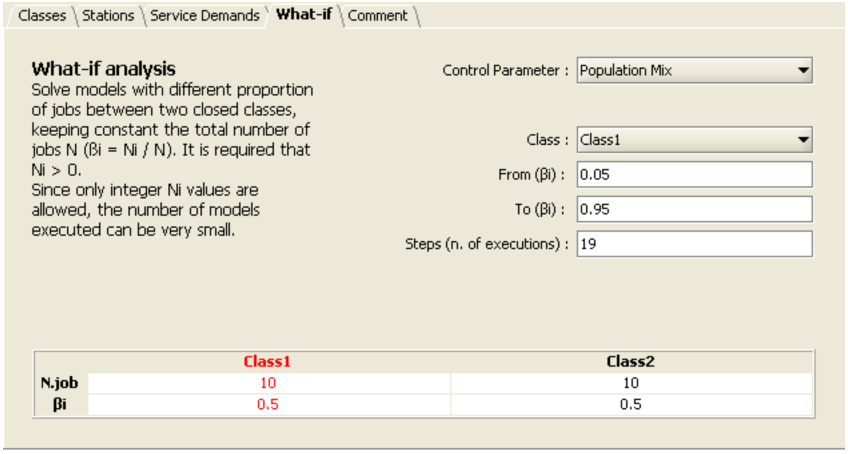


Service Demands

Input service demands of each station and class.
If the station is "Load Dependent" you can set the service demands for each number of customers by double-click on "LD Settings..." button.
Press "Service Times and Visits" button to enter service times and visits instead of service demands.

*	Class1	Class2
Station1	1.000000	5.000000
Station2	5.000000	1.000000
Station3	1.000000	5.000000

Figure 2.42: Example 5 - Service Demands Tab



What-if analysis
Solve models with different proportion of jobs between two closed classes, keeping constant the total number of jobs N (βi = Ni / N). It is required that Ni > 0.
Since only integer Ni values are allowed, the number of models executed can be very small.

Control Parameter : Population Mix

Class : Class1

From (βi) : 0.05

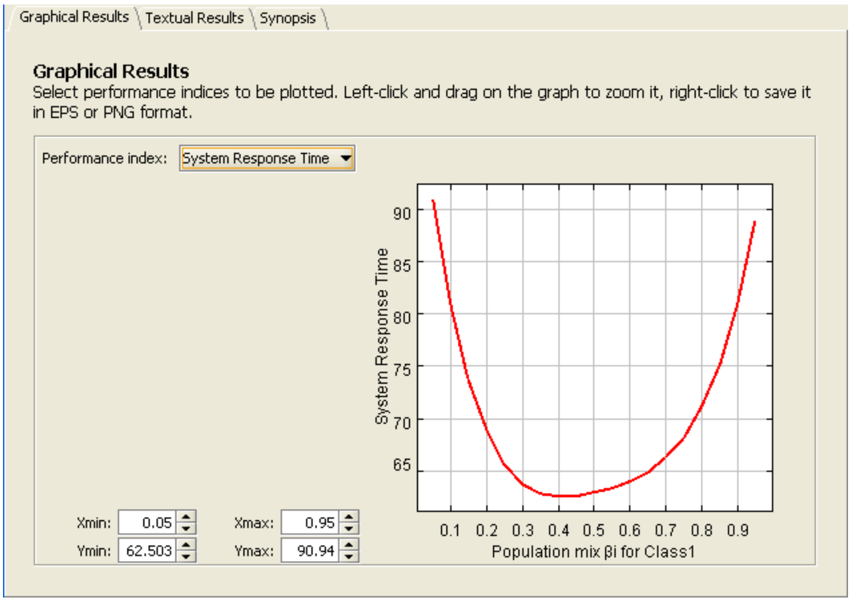
To (βi) : 0.95

Steps (n. of executions) : 19

	Class1	Class2
N.job	10	10
βi	0.5	0.5

Figure 2.43: Example 5 - What-if Tab

value of System Throughput (0.32 job/s) and the minimum value of System Response Time (62.50 s). They both corresponds to the execution with population mix $\beta_1 = 0.40$ for *Class1*, which means that the optimal population values are $N_1 = 8$ and $N_2 = 12$.



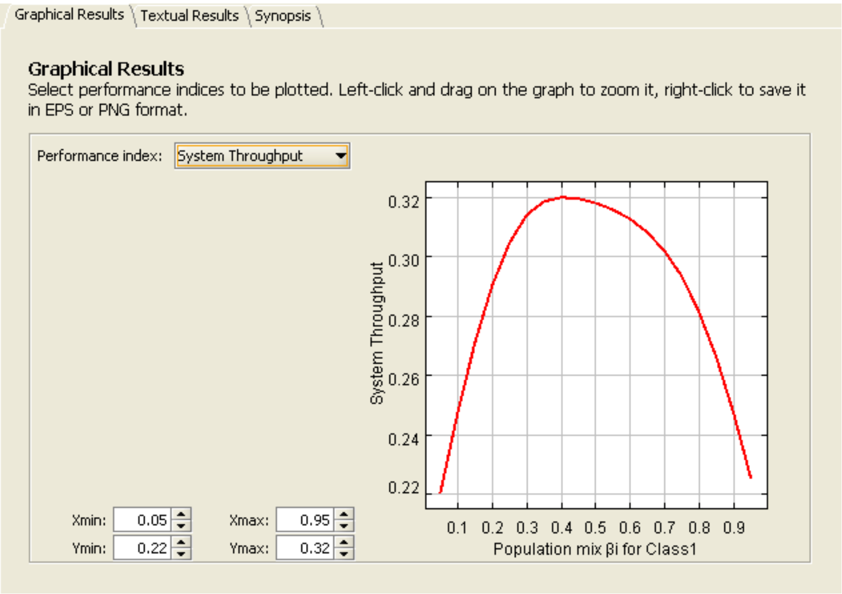
Graphical Results
Select performance indices to be plotted. Left-click and drag on the graph to zoom it, right-click to save it in EPS or PNG format.

Performance index: System Response Time

Xmin: 0.05 Xmax: 0.95

Ymin: 62.503 Ymax: 90.94

Figure 2.44: Example 5 - System Response Time



Graphical Results

Select performance indices to be plotted. Left-click and drag on the graph to zoom it, right-click to save it in EPS or PNG format.

Performance index: System Throughput

Xmin: 0.05
Xmax: 0.95
Ymin: 0.22
Ymax: 0.32

Figure 2.45: Example 5 - System Throughput

Chapter 3

JABA

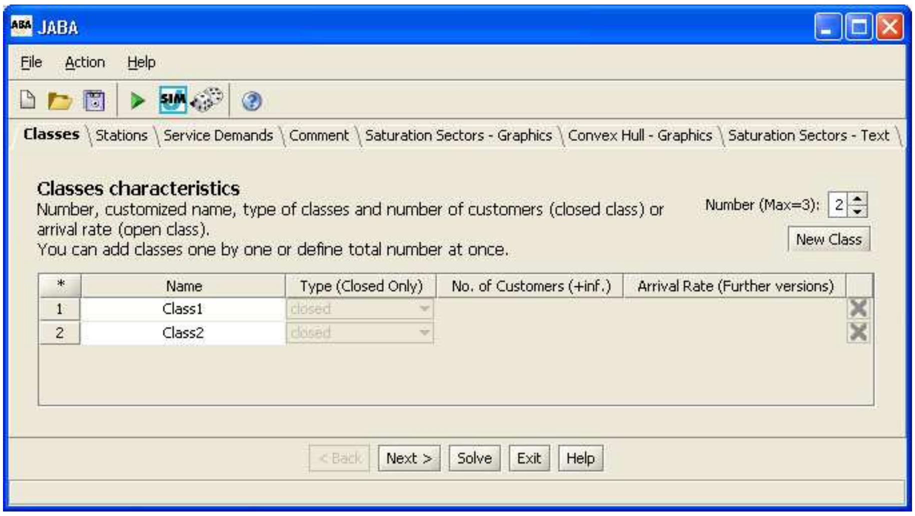
3.1 Overview

Product-form queueing network models have been used for modelling the performance of many type of systems, from large computing infrastructures to distributed applications. The complexity of modern systems makes the application of known exact solution techniques, such as the convolution algorithm or the MVA prohibitively expensive in terms of computational resource requirements. Also approximate solution techniques become less accurate or more expensive with the growing complexity of the model. The alternative is represented by asymptotic techniques that can efficiently determine asymptotic values for several performance indices such as throughput, response time and queue lengths. The asymptotic techniques are particularly useful in tuning studies where one needs to evaluate the performance gains of different tuning alternatives. The key to determine the asymptotic performances is the knowledge of the queueing center(s) with the highest utilization, i.e. the bottleneck station(s). Multiclass models, and in particular two class models which are widely-used, can exhibit multiple simultaneous bottlenecks depending of the population mix. While identifying the bottleneck stations under a single-class workload is a well-established practice, no simple methodology for multiclass models has yet been found. JABA provides such a technique, called Polyhedral Analysis, using a convex polytopes based approach presented in Casale and Serazzi, 2004 [GC04].

Among the no-bottleneck station, we distinguish two different centers: dominated and masked-off stations. This distinction is important since the masked-off stations, despite having lower utilization than the bottleneck center(s), may still exhibit the largest queue-length and hence the highest response times. Dominated stations, instead, typically play a marginal role in determining system performance, and have always the smallest utilization and queue-lengths.

3.1.1 Starting the alphanumeric JABA solver

Selecting ABA button on the starting screen, Figure 3.1 the JABA window shows up.



Classes characteristics

Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

*	Name	Type (Closed Only)	No. of Customers (+inf.)	Arrival Rate (Further versions)
1	Class1	closed		
2	Class2	closed		

Figure 3.1: Classes tab

Three main areas are shown:

Menu : it is organized into three groups of functions. To use a menu, click on the menu heading and choose the appropriate option. For the description of menu entries, see section 3.3

Toolbar : contains some buttons to speed up access to JABA functions (e.g. New model, Open, Save. . . See section 3.3 for details). If you move the mouse pointer over a button a tooltip will be shown up.

Page Area : this is the core of the window. All JABA parameters are grouped in different tabs. You can modify only a subset of them by selecting the proper tab, as will be shown later.

3.2 Model definition

Models with two or three customer classes provide estimates of wich stations are potential bottleneck. For a brief description of basic terminology please refer to Appendix A.

The workload is characterized for each station by the service demands of every customer class. Al least JABA works with 2 classes of custumers therefore a matrix of service demands is required [LZGS84].

3.2.1 Defining a new model

To define a new model select toolbar button 🗋 or the *New* command from *File* menu. The following parameters must be defined:

1. `Classes`
2. `Stations` (service centers)
3. `Service demands` (or Service Times and Visits)
4. Optional short `Comment`

The execution of JABA produces the following graph:

- Saturation Sector - Graphics
- Convex Hull - Graphics

And a textual report about the saturation sectors:

- Saturation Sectors - Text

Input tabs

As shown in Figure 3.1, the parameters that must be entered in order to define a new model are divided in four tabs: `Classes`, `Stations`, `Service Demands` and `Comment`.

The number of tabs become five, if you click *Service Times and Visits* button in `Service Demands Tab`. As will be discussed in subsection 3.2.4, the `Service Demands Tab` will be hidden and it will appear `Service Times Tab` and `Visit Tab`. You can navigate across tabs:

- using sequential wizard buttons, if enabled, at the bottom of the window (Figure 3.2)
- using sequential buttons located in menu
- using the tab selector, clicking on the corresponding tab (Figure 3.3)

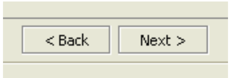


Figure 3.2: Wizard buttons

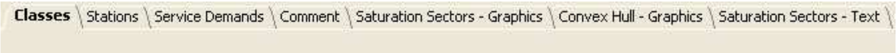
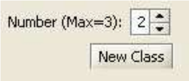


Figure 3.3: Tab selector

3.2.2 Classes Tab

An example screenshot of this tab can be seen in Figure 3.1. This tab allows to set the number of custumer classes. It's possible to set two or three clesses.

The number of classes in the model can be specified in the corresponding input area, shown in Figure 3.4 and can be modified using the keyboard or the spin controls.



Number (Max=3): 2

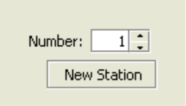
Figure 3.4: Number of classes

Using the delete button ☒ associated to a specific class, a class can be removed provided that there will be at least one class after the deletion. Similar result may be obtained using spin controls, decreasing classes number; in this case last inserted class will be removed.

Default class names are *Class1*, *Class2*, ... *ClassN*. The model can be personalized by changing these names.

3.2.3 Stations Tab

The number of stations of the model can be specified in the corresponding input area (Figure 3.5) and can be modified using the keyboard or the spin controls.



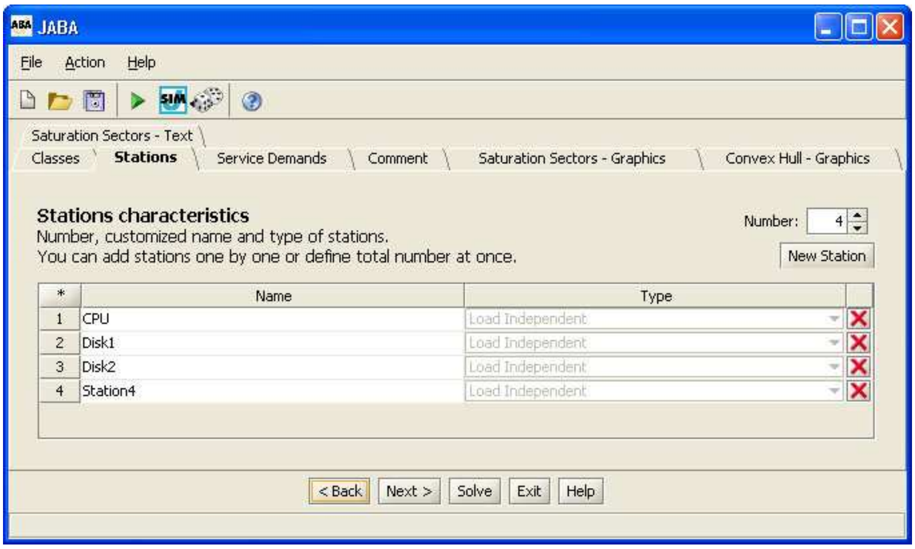
Number: 1

Figure 3.5: Number of stations

Using the delete button ☒ associated to a specific station, a station can be removed provided that there will be at least one station after the deletion. Similar result may be obtained decreasing stations number using spin controls; in this case last inserted station will be removed.

Default station names are *Station1*, *Station2*, ... *StationN*. In order to personalize your model, you can change and give names other than default ones.

In Figure 3.6 there is only one station with default name *Station4* and there are three stations with personalized names: *CPU*, *Disk1* and *Disk2*.



Stations characteristics
Number, customized name and type of stations.
You can add stations one by one or define total number at once.

Number: 4

*	Name	Type
1	CPU	Load Independent
2	Disk1	Load Independent
3	Disk2	Load Independent
4	Station4	Load Independent

Figure 3.6: Defining the stations name

3.2.4 Service Demands, Service Times and Visits Tabs

Service Demands can be defined in two ways:

- directly, by entering Service Demands (D_{kc})

- indirectly, by entering Service Times (S_{kc}) and Visits (V_{kc})

Service demand D_{kc} is the total service requirement, that is the average amount of time that a customer of class c spends in service at station k during one interaction with the system, i.e. it's complete execution. Service time S_{kc} is the average time spent by a customer of class c at station k for a single visit at that station while V_{kc} is the average number of visits at that resource for each interaction with the system.

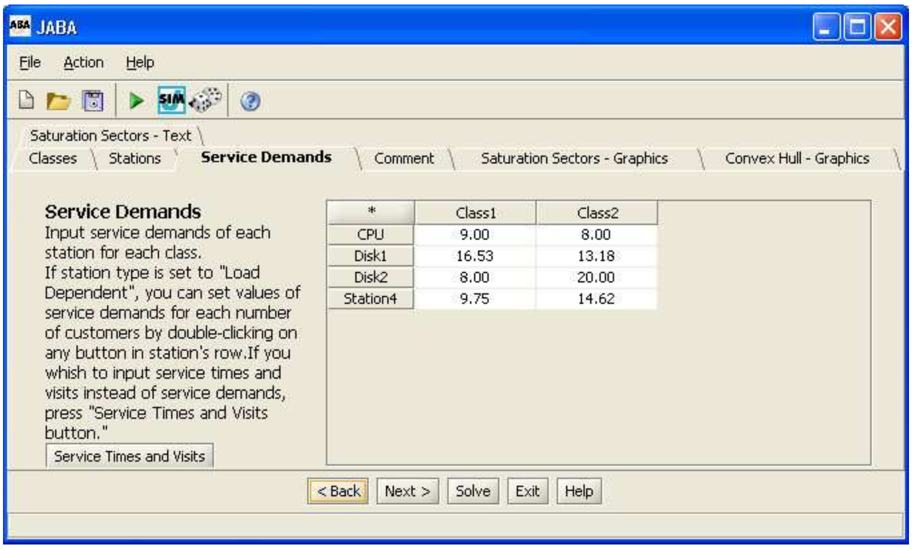
Remember that $D_{kc} = V_{kc} * S_{kc}$ so it's simple to compute service demands matrix starting from service times and visits matrixes. Inverse calculation is performed with the following algorithm:

$$V_{kj} = \begin{cases} 1 & \text{if} \quad D_{kc} > 0 \\ 0 & \text{if} \quad D_{kc} = 0 \end{cases}$$

$$S_{kc} = \begin{cases} D_{kc} & \text{if} \quad D_{kc} > 0 \\ 0 & \text{if} \quad D_{kc} = 0 \end{cases}$$

Service Demands Tab

In this tab you can insert directly Service Demands D_{kc} for each pair {station k-class c} in the model. Figure 3.7 shown a reference screenshot can be seen: notice that a value for every D_{kc} element of the D-matrix has already been specified because the default value assigned to newly created stations is zero.



Service Demands

Input service demands of each station for each class.
If station type is set to "Load Dependent", you can set values of service demands for each number of customers by double-clicking on any button in station's row.If you whish to input service times and visits instead of service demands, press "Service Times and Visits button."

*	Class1	Class2
CPU	9.00	8.00
Disk1	16.53	13.18
Disk2	8.00	20.00
Station4	9.75	14.62

Figure 3.7: The Service Demands Tab

In Figure 3.7, each job of *Class1* requires an average service demand time of 6 sec to *CPU*, 16.53 sec to *Disk1*, 8 sec to *Disk2* and 9.75 sec to *Station4*.

Service Times and Visits Tabs

In the former tab you can insert the Service Times S_{kc} for each pair {station k-class c} in the model, in the latter you can enter the visits number V_{kc} (See Figure 3.8).

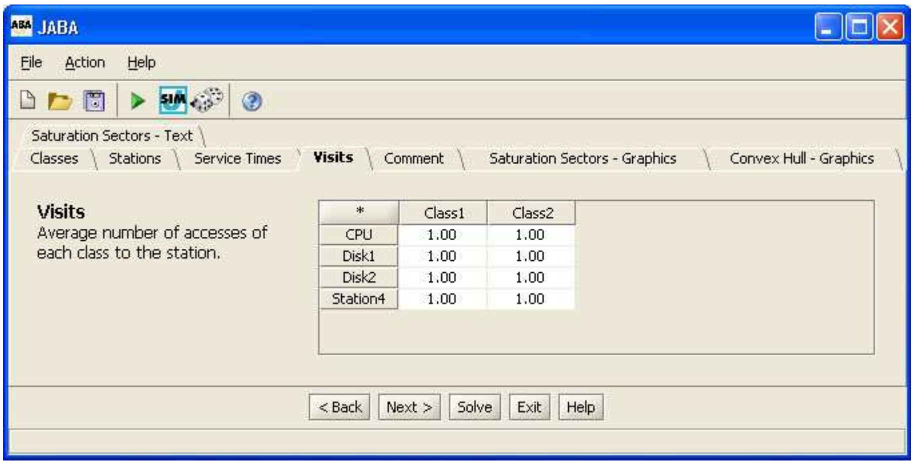
The layout of these tabs is similar to the one of the `Service Demand Tab` described in the previous paragraph. The default value for each element of the Service Times (S) matrix is zero, while it's one for Visits' matrix elements.

3.2.5 Comment Tab

In this Tab, a short - optional - comment about the model can be inserted; it will be saved with the other model parameters.

3.2.6 Saturaction Sector - Graphic

Using the `Solve` command the Saturaction Sector graphic appears.



Visits
Average number of accesses of each class to the station.

*	Class1	Class2
CPU	1.00	1.00
Disk1	1.00	1.00
Disk2	1.00	1.00
Station4	1.00	1.00

Figure 3.8: Visits Tab

Two Classes graphic

We indicate a certain mix population with the couple (N1,N2) where N1 is the percentage of the customers belong to class1 and N2 is the percentage of the customers of class2. Because the sum of N1 and N2 must be 100% we will only consider N1 (N2 can be calculated subtrahend N1 at 100). In the hypothesis that the number of customers in the system is constant (and high) it is possible to find an interval [N1 to N1'] where the same two stations are bottleneck. We define this interval saturation sector and its extremity switching points that are the value in with the station change from non-bottleneck to bottleneck or viceversa.

It is possible to represent a saturation sector and the switching points using a bidimentional graphic Figure 3.9.

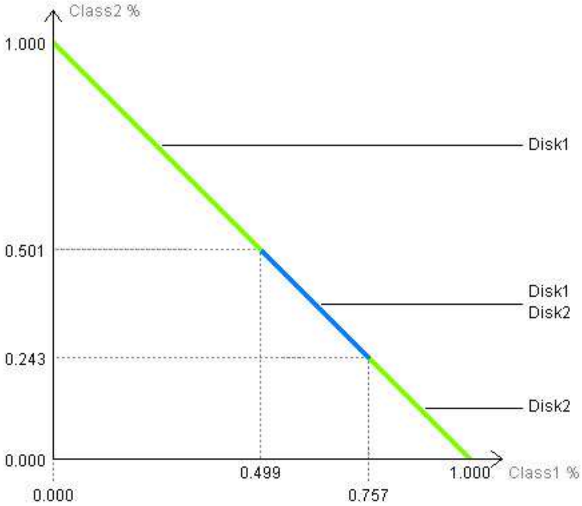


Figure 3.9: Saturation Sector Graphic

In this example we can observe that *Disk1* is a bottleneck for a mix population among (0;1) and (0.499,0.501). At the point (0.499,0.501) color switches from green to blue because this point is a switching point. In fact for a mix population from (0.499,0.501) to (0.757,0.243) *Disk1* and *Disk2* are both bottleneck. After the point (0.757,0.243), that is a swiching point too, only *Disk2* is a bottleneck.

Three Classes graphic

Suppose we can split the customers of our system in 3 class of customers, a specific population mix could be represented by a point whose coordinates are the percentage of customers of a specific class. on a triangle that has the vertex in the points (1,0,0), (0,1,0) e (0,0,1). The triangle area represent every possible population mix. So we can divide the triangle in different sub-areas and each area rapresent a population mix range where one, two or more stations are bottleneck at the same time. It is clearly understandable that we have no more a switching point (such as in the two classes graph) but a switching segment.

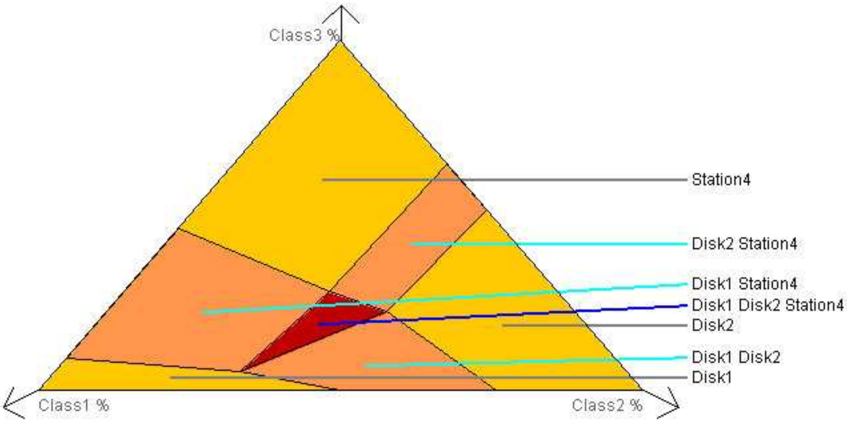


Figure 3.10: Saturation Sector Graphic

In this example Figure 3.10 we can observe a big area,on the top of the triangle, that rapresents the range of mix population where *Station4* is a bottleneck. In the center of the triangle there is a little red area that rapresent the range of mix population where *Disk1*, *Disk2* and *Station4* are all bottleneck.

This graph is useful for understanding what station begin bottlenek and when but it is not simple to recornize the exat value of the swiching segment. In order to obtain the exact value select the “Saturation Sector - Text” tab subsection 3.2.8.

3.2.7 Convex Hull - Graphic

Using the `Solve` command and selecting the “Convex Hull - Graphic” tab the Convex Hull graphic is show.

Overview

The Convex Hull graphic implements the polyhedral analysis technique. The polyhedral analysis technique implemented in JABA is limited to queueing networks with customers grouped in two classes. The points in the graph are obtained from the service demand matrix as follows. Each station corresponds to a point on the graph whose co-ordinates are represented by the service demand of the two customer classes. For example, a queue with service demands 10 sec for class 1, and 15 sec for class 2 is represented as a point with coordinates (10,15). Then JABA builds the convex hull of this set of points, that is the smallest convex set containing all the points.

The distinction of queues among the different types of bottleneck and non-bottleneck classes can be immediately operated from the knowledge of the convex hull using the following rules:

- All *potential bottlenecks* lie on the boundary of the convex polytope.
- All *dominated* station are interior points P of the convex polytope such that there is at least one point Q whose co-ordinates are all higher or equal in value than those of P;
- All non-dominated stations in the interior of the convex polytope are *masked-off* stations. The application of these rules is described below in the case study.

From this plot,Figure 3.11 we can see that *Disk1* and *Disk2* (depicted in red) lie on the boundary of the convex hull, thus are potential bottlenecks. *CPU* (green) is instead a dominated non-bottleneck station, since it is dominated by *Sation4* and *Disk1*. *Sation4* (blue) is also in the interior of the convex hull, but it is a non-bottleneck station as there is no other point with all coordinates greater than those of *Sation4* so it is a masked-off station.

Determining points coordinates

In order to gain exact information about the coordinates of a point it is sufficient to click, using the left button of the mouse, on the point and additional information will appear after the point’s name. If the clicked point is a dominated one, then it will be possible to see by which point (there could be more than one) it is dominated.

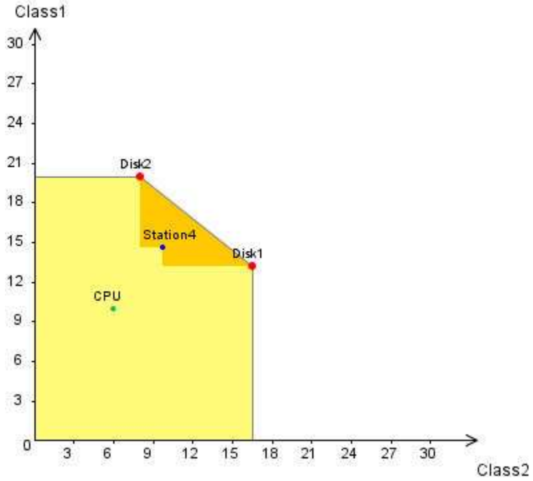


Figure 3.11: Convex Hull Graphic

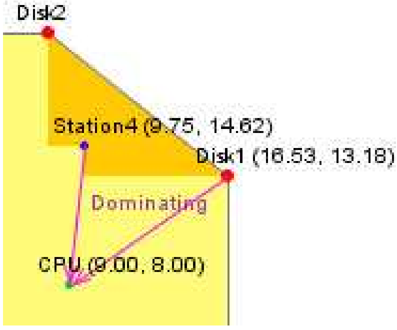


Figure 3.12: Convex Hull Graphic while CPU is selected

Determining heavy-load throughputs and bottlenecks under a certain workload mix

Clicking (left button) on the line that it connects two bottleneck stations you can obtain information about the saturation sector such as the interval of mix population that make possible the bottleneck and also the throughput of the two classes.

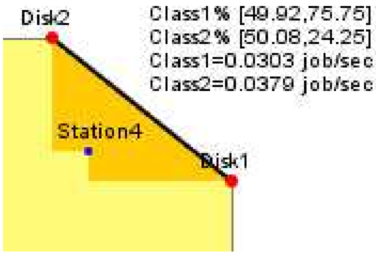


Figure 3.13: Convex Hull Graphic while saturation sector is selected

Move a station to another point

Now suppose that we have to analyze the upgrade of the station *Station4*, the new *Station4* will have half demand for both the customer's classes compared to the old one. In order to do this we can change the value in the demand's table of JABA or click on the point (left button) and drag it in the new position.

While dragging the point temporary co-ordinates are shown and when the point is released all data are update and a new graph is created.

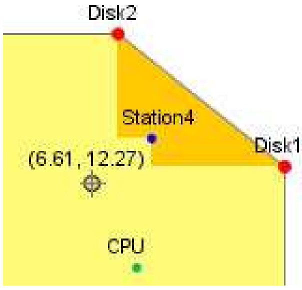


Figure 3.14: Convex Hull Graphic while is dragging a point

Identify a point's label in a crowded area

It is possible that a area of the graph could be crowded of points whose names are difficult to read as showed in thefollowing graph Figure 3.15.

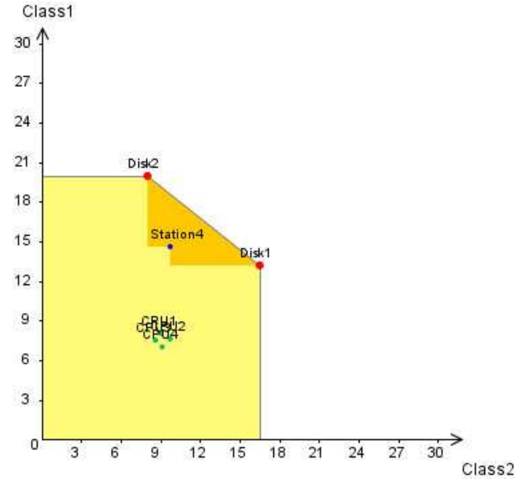


Figure 3.15: Convex Hull Graphic whith a crowded area

It is difficult to understand - or simply read - which of the dominated point the label belongs to; at first we can try to change the zoom, we can therefore zooming in by a double click on the mouse's left button or zooming out by a double click on the mouse's right button. If the zooming is no enough we can try filtering the zone and freeing only some points. In order to filter the area is enough to select it keeping the mouse's left button pressed. The background of the selected area will become grey, points will become dark grey and their labels will disappear.

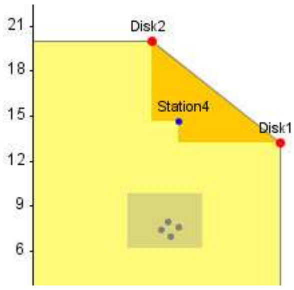


Figure 3.16: Convex Hull Graphic whith a filtered area

Now we try to reduce the filtered area adding some free area; to do that we select the area that we wish to free keeping the mouse's right button pressed. In the following example we have set the tallest point free.

Only the name of the station that is in the free area is showed.

3.2.8 Saturation Sector - Text

It's a simple report about the saturation sector, for each saturation sector you can find:

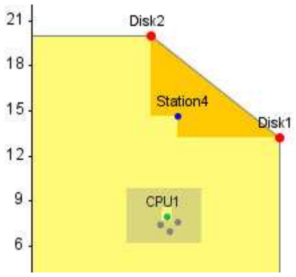


Figure 3.17: Convex Hull Graphic whith a filtered area

- The station that are bottleneck
- The switching point or the switching segment bound points.

This tab could be helpful in the Saturation Sector graphic of three classes of costumers.subsection 3.2.6

3.2.9 Modification of a model

To modify system parameters return to the main window and enter new data or moving a station in Convex Hull - graphic tab (see subsection 3.2.7). After the modifications, if you use `Solve` command, the new graphs will show. You can `save` this new model with the previous name - overwriting the previous one - or `save` it with a different name or in a different directory.

3.3 Menu entries

3.3.1 File

New

Use this command in order to create a new JABA model.

Shortcut on Toolbar:	🗋
Accelerator Key:	CTRL+N

Open

Use this command to open an existing model. You can only open one model at time, to edit two or more models start more than one instance of JABA. If current model was modified since its creation or last save action, a popup window will be shown for confirmation.

It's possible to open not only models saved with JABA (*.jaba), but also with other programs of the suite (for example JMVA *.jmva, JSIM *.jsim and JMODEL *.jmodel). Whenever a foreign data file is opened, a conversion is performed and error/warnings occurred during conversion will be reported in a window.

Models are stored in XML format, see *JMT system manual* for a detailed description.

Shortcut on Toolbar:	📂
Accelerator Key:	CTRL+O

Save

Use this command in order to save the active document with its current name in the selected directory.

When you save a document for the first time, JABA displays the Save As dialog box so you can rename your document. If you save a model after its resolution, results are stored with model definition data.

Shortcut on Toolbar:	💾
Accelerator Key:	CTRL+S

Exit

Use this command in order to end a JABA session. You can also use the Close command on the application Control menu. If current model was modified since its creation or last save action, a popup window will be shown for confirmation.

Accelerator Key:	CTRL+Q

3.3.2 Action

Solve

Use this command when model description is terminated and you want to start the solution of the model. At the end of the process the Saturation Sector - Graphic subsection 3.2.6 will show.

Shortcut on Toolbar:	▶
Accelerator Key:	CTRL+L

Randomize

Use this command in order to insert random values into Service Demands - or Service Times - table.

Shortcut on Toolbar:	🎲
Accelerator Key:	CTRL+R

3.3.3 Help

JABA Help

Not still avaiable.

3.4 Examples

In this section we will describe some examples of model parametrization and solution using JABA exact solver. Step-by-step instructions are provided in two examples:

1. A two class model (subsection 3.4.1)
2. A three class model (subsection 3.4.2)

3.4.1 Example 1 - A two class model

Solve the two class model specified in Figure 3.18.

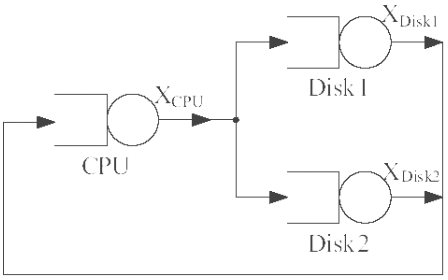


Figure 3.18: Example 1 - network topology

There are three stations (named *CPU*, *Disk1* and *Disk2*). Service demands for each station are reported in Table 3.1.

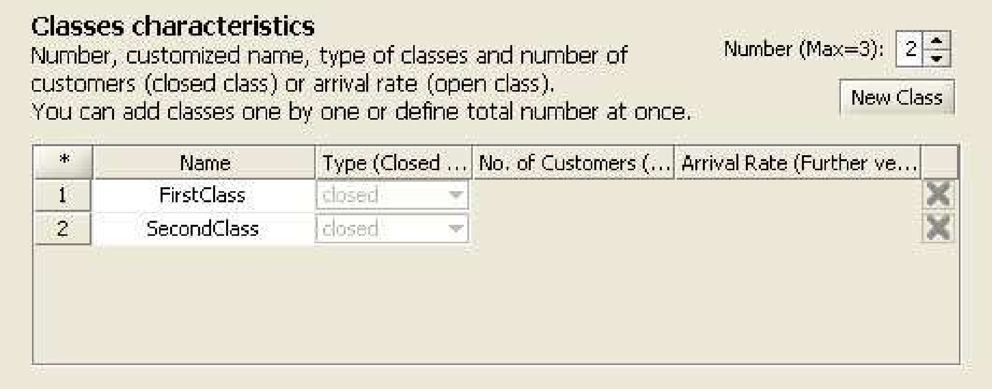
	CPU	Disk1	Disk2
FirstClass	2.60	7.02	4.81
SecondClass	7.02	4.51	5.70

Table 3.1: Example 1 - service demand

Step 1 - Classes Tab

- use New command to create a new JABA document
- by default, you have already two classes
- if you like, substitute default *Class1* and *Class2* name with a customized one (*FirstClass* and *SecondClass* in our example)

At the end of this step, the `Classes Tab` should look like Figure 3.19.



Classes characteristics
Number, customized name, type of classes and number of customers (closed class) or arrival rate (open class).
You can add classes one by one or define total number at once.

Number (Max=3): 2

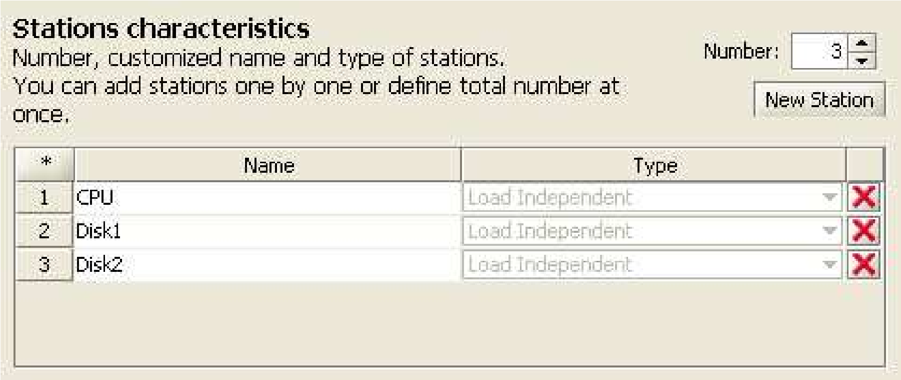
*	Name	Type (Closed ...	No. of Customers (...	Arrival Rate (Further ve...	
1	FirstClass	closed			✕
2	SecondClass	closed			✕

Figure 3.19: Example 1 - input data (Classes Tab)

Step 2 - Stations Tab

- use `Next` > command to switch to `Stations Tab`
- digit number 3 into stations number textbox or select number 3 using spin controls or push *New Station* button twice. Now your model has three stations with a default name
- if you want you can change station names. Substitute *CPU* for default name *Station1*, substitute *Disk1* for default name *Station2*, substitute *Disk2* for default name *Station3*

At the end of this step, the `Stations Tab` should look like Figure 3.20.



Stations characteristics
Number, customized name and type of stations.
You can add stations one by one or define total number at once.

Number: 3

*	Name	Type	
1	CPU	Load Independent	✕
2	Disk1	Load Independent	✕
3	Disk2	Load Independent	✕

Figure 3.20: Example 1 - input data (Stations Tab)

Step 3 - Service Demand Tabs

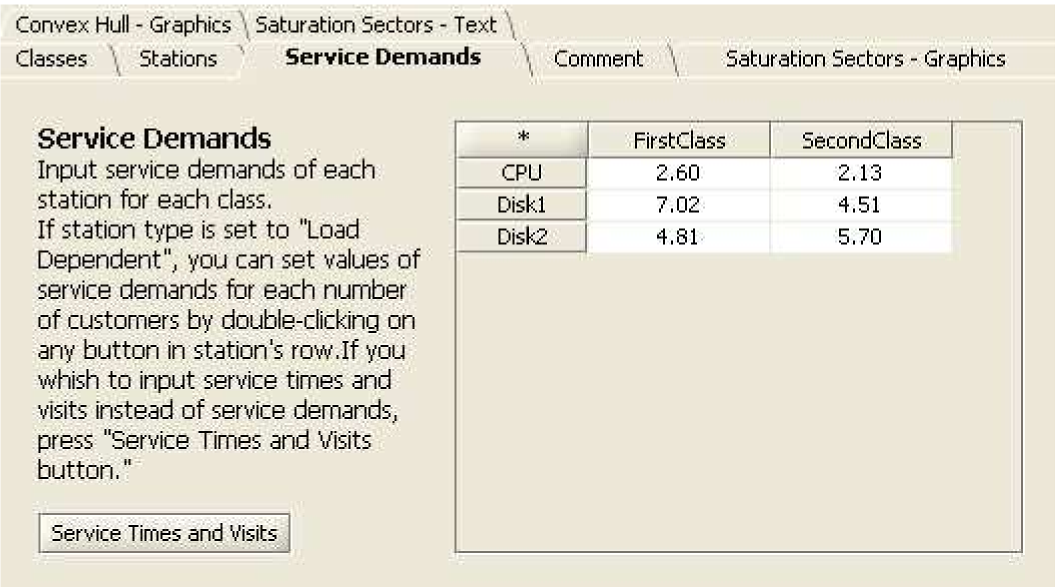
- use `Next` > command to switch to `Service Demands Tab`
- you can input all Demands in the table.
- press *Service Time and Visit* button if you don't know the Service Demands of the three stations. After button pressure, the `Service Demands Tab` will be hidden and `Service times Tab` and `Visit Tab` will appear. Now Service Times and number of Visits should be typed.

At this point, the `Service Demands Tab` should look like Figure 3.21.

Step 4 - Saturation Sectors - Graphics

Use `Solve` command to start the solution of input model. In the *Saturation Sector - Grlaphic Tab* will be displayed a graphic like the one of Figure 3.22.

This graphic show the range of mix population where two or more stations are both bottleneck. In this example *Disk1* and *Disk2* are both bottleneck when *FirstClass* is among *0.545 - 0.689* and *SecondClass* is among *0.311 - 0.455*



Service Demands

Input service demands of each station for each class.
If station type is set to "Load Dependent", you can set values of service demands for each number of customers by double-clicking on any button in station's row.If you whish to input service times and visits instead of service demands, press "Service Times and Visits button."

*	FirstClass	SecondClass
CPU	2.60	2.13
Disk1	7.02	4.51
Disk2	4.81	5.70

Figure 3.21: Example 1 - input data (Service Demand Tab)

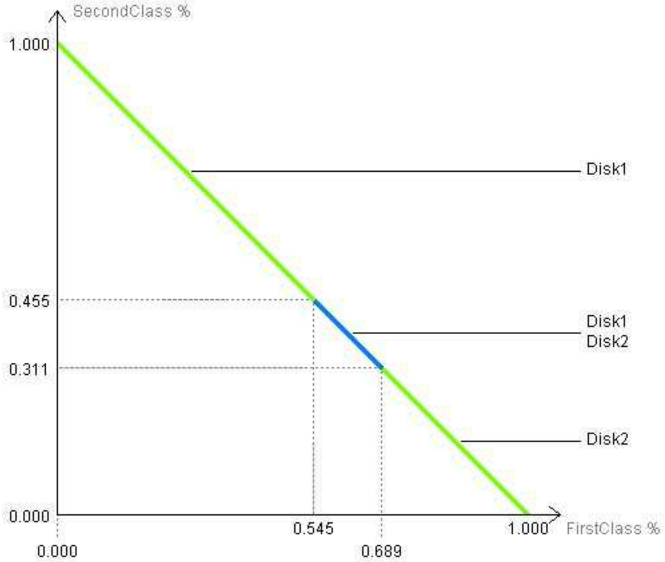


Figure 3.22: Example 1 - Saturation Sectors Graphics

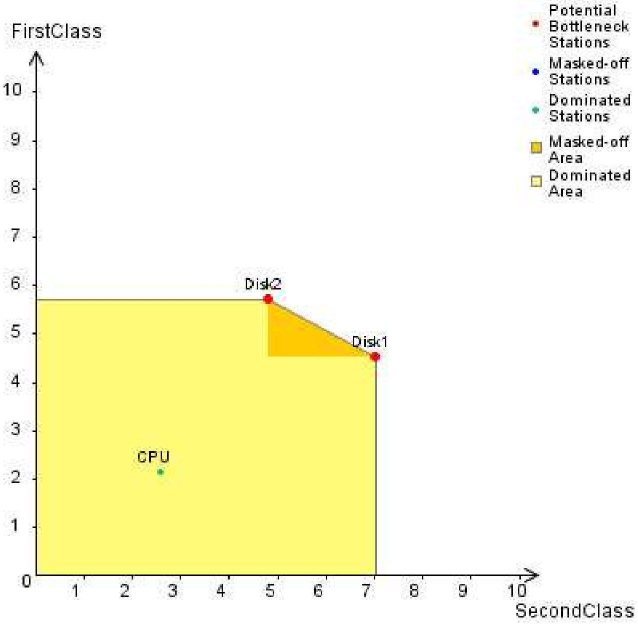


Figure 3.23: Example 1 - Convex Hull - Graphics

Step 5 - Convex Hull - Graphics

Using `Next` > command to switch to `Convex Hull - Graphics` and a graphic like Figure 3.23 will be shown.

This graphic implements the polyhedra analysis technique. The points on this graphic represent the stations. Dominated stations are green, masked-off ones are blue and bottlenek ones are red. If you click on a station, additional information will be shown, otherwise you can drag a station and see what will change in your model.

Step 6 - Saturation Sectors - Text

If you use `Next` > command again you will see the *Saturation Sectors - Text*, a simple textual report about the Saturation Sectors.

3.4.2 Example 2 - A three class model

Solve the multiclass open model specified in Figure 3.24. The model is characterized by three A , B and C.

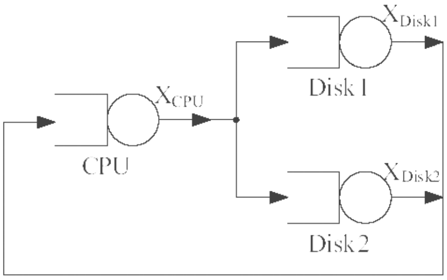


Figure 3.24: Example 2 - network topology

There are three stations, identified with names *CPU*, *Disk1* and *Disk2*. Service times and visits for stations are shown in Table 3.2 and Table 3.3.

	CPU	Disk1	Disk2
Class A [s]	0.01	0.38	0.30
Class B [s]	0.02	0.62	0.80
Class C [s]	0.09	0.42	0.50

Table 3.2: Example 2 - service times

	CPU	Disk1	Disk2
Class *A*	101.0	60.0	40.0
Class *B*	44.0	16.0	27.0
Class *C*	63.0	21.0	35.0

Table 3.3: Example 2 - number of visits

Step 1 - Classes Tab

It is the same procedure of the Example 1 (subsection 3.4.1). You have only to add one more class.

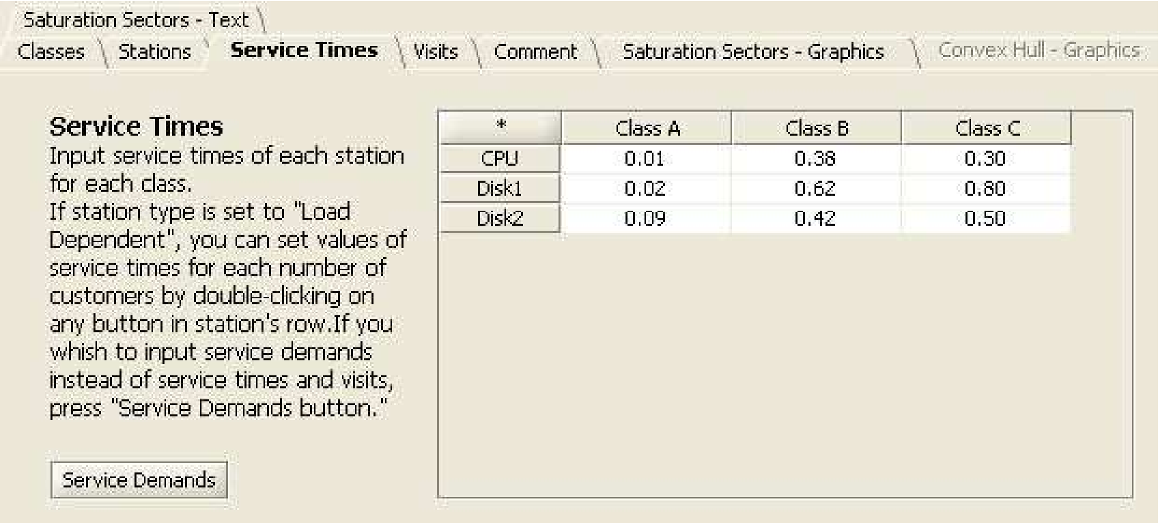
Step 2 - Stations Tab

It is the same procedure of the Example 1 (subsection 3.4.1).

Step 3 - Service Times and Visits Tabs

- use `Next` > command to switch to `Service Demands Tab`
- press *Service Time and Visit* button if you don't know the Service Demands of the three stations, Now Service Times and number of Visits should be typed. After button pressure, the `Service Demands Tab` will be hidden and `Service times Tab` and `Visit Tab` will appear
- you can input all Service Times in the table.

At this point, the `Service Times Tab` should look like Figure 3.25.



Service Times

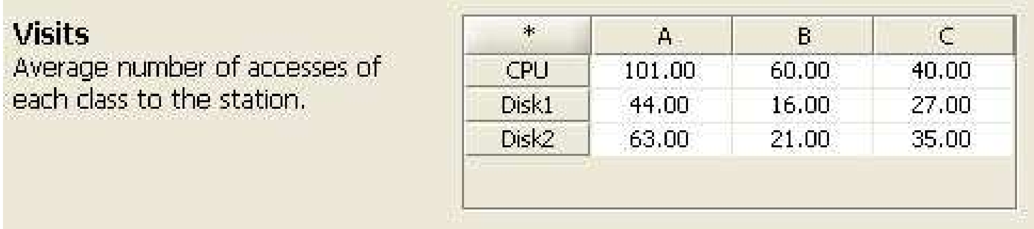
Input service times of each station for each class.
If station type is set to "Load Dependent", you can set values of service times for each number of customers by double-clicking on any button in station's row.If you whish to input service demands instead of service times and visits, press "Service Demands button."

*	Class A	Class B	Class C
CPU	0.01	0.38	0.30
Disk1	0.02	0.62	0.80
Disk2	0.09	0.42	0.50

Figure 3.25: Example 2 - input data (Service Times Tab)

- use `Next` > command to switch to `Visits Tab`
- input number of visits for each center in the table.

At the end of this step, the `Visits Tab` looks like Figure 3.26.



Visits

Average number of accesses of each class to the station.

*	A	B	C
CPU	101.00	60.00	40.00
Disk1	44.00	16.00	27.00
Disk2	63.00	21.00	35.00

Figure 3.26: Example 2 - input data (Visits Tab)

Step 4 - Saturation Sectors - Graphics

Use `Solve` command to start the solution of the input model. In the *Saturation Sector - Graphic Tab* will be displayed a graphic like the one of Figure 3.27.

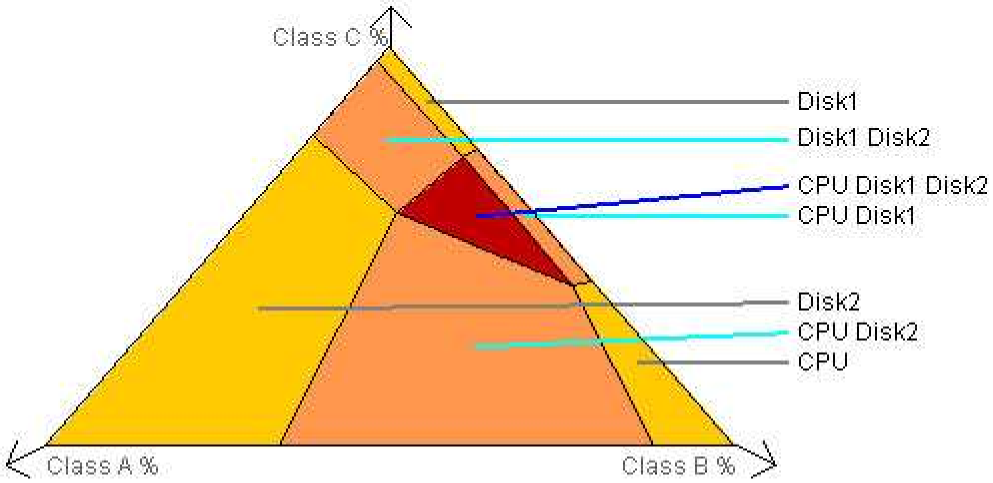


Figure 3.27: Example 2 - Saturation Sectors Graphics

This graphic show the range of mix population where two or more stations are both bottleneck. For example the orange area is a renge of mix population in which *Disk1* and *Disk2* are both bottlenek.

Step 5 - Saturation Sectors - Text

If you use `Next` > command again you will see the *Saturation Sectors - Text*, a simple textual report about the Saturation Sectors.

Chapter 4

JWAT - Workload Analyzer Tool

A computer system can be viewed as a set of hardware and software resources that are used in a time-variant fashion by the processing requests generated by the user. During any given time interval, the users submit their processing requests to the systems by inputting sets of programs, data, commands, requests for web sites or file downloads, etc. All this input requests are usually designated by the collective term of *workload*. Every time the value of a system or network performance index is given, the workload under which that values obtained must be reported or, at any rate, known, since performance cannot be expressed by quantities independent of the system's workload. Thus, no evaluation problem can be adequately solved if the workloads to be processed by the systems or transmitted over a network are not specified.

The quantitative description of a workload's characteristics is commonly called *workload characterization*. A characterization is usually done in terms of workload parameters that can affect system behavior and are defined in a form suitable for a desired use. Thus, a workload is characterized correctly if, and only if, its result is a set of quantitative parameters selected according to the goals of the characterization operation. Workload characterization is fundamentally important in all performance evaluation studies and is indispensable for designing useful workload models.

A workload may be regarded as a set of vectors in a space with a number of dimensions equal to the number of the workload parameters used to describe each element considered. Since the amount of resulting data is generally considerable, their analysis for the purpose of identifying groups of components with similar characteristics will have to be performed by statistical techniques that can handle multidimensional samples, that is, techniques that deal with multiple factors.

The JWAT tool is based on the application of a set of statistical techniques oriented towards the characterization of log data representing resource utilizations, traffic of requests, user web paths, etc. Its application provides the input data for the models to be used in all types of performance evaluation studies.

The main window of JWAT (Figure 4.1) is opened pressing the button in the JMT suite main window. Through this window, it is possible to select one of the three main components of JWAT:

- **Traffic Analysis:** Characterization of the arrival traffic of requests analyzing the timestamp in a log file and deriving the parameters that refer to the burstiness of requests

- **Workload Analysis:** Characterization of the workload of a system through the analysis of the log files applying a clustering algorithm (k-Means or Fuzzy k-Means) and other statistical techniques. The input data can be loaded directly from standard log files (e.g., Apache log files, IIS) or from files with any format described by the users. Several graphical representations of the results are also provided for their statistical comparison (histograms, pie-charts, dispersion matrix, QQ-Plot, QQ-Plot matrix, scatter plots, ...).

- **Path Analysis:** The navigation paths followed by the users of a web site allow the identification of the access paths that should be considered as representatives of the workload analyzed.

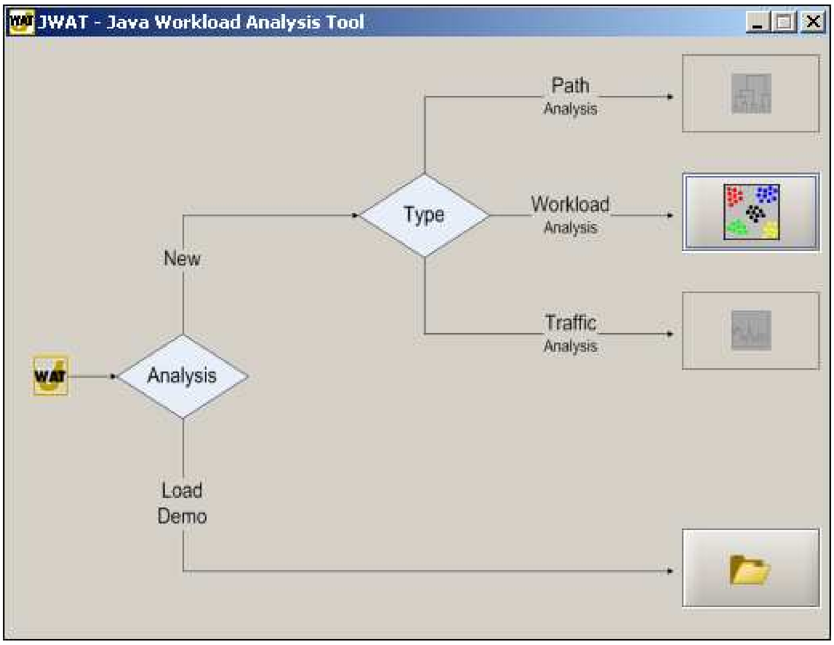


Figure 4.1: Main window of JWAT - Workload Analyzer Tool

4.1 Workload analysis

To perform a workload analysis press the button of the JWAT window and the window of Figure 4.2 will be opened. This application help the user to perform in the proper sequence the various steps of the workload characterization from the data loading to the statistical analysis, to the results visualization and to their evaluation. The main components of the workload analysis window are:

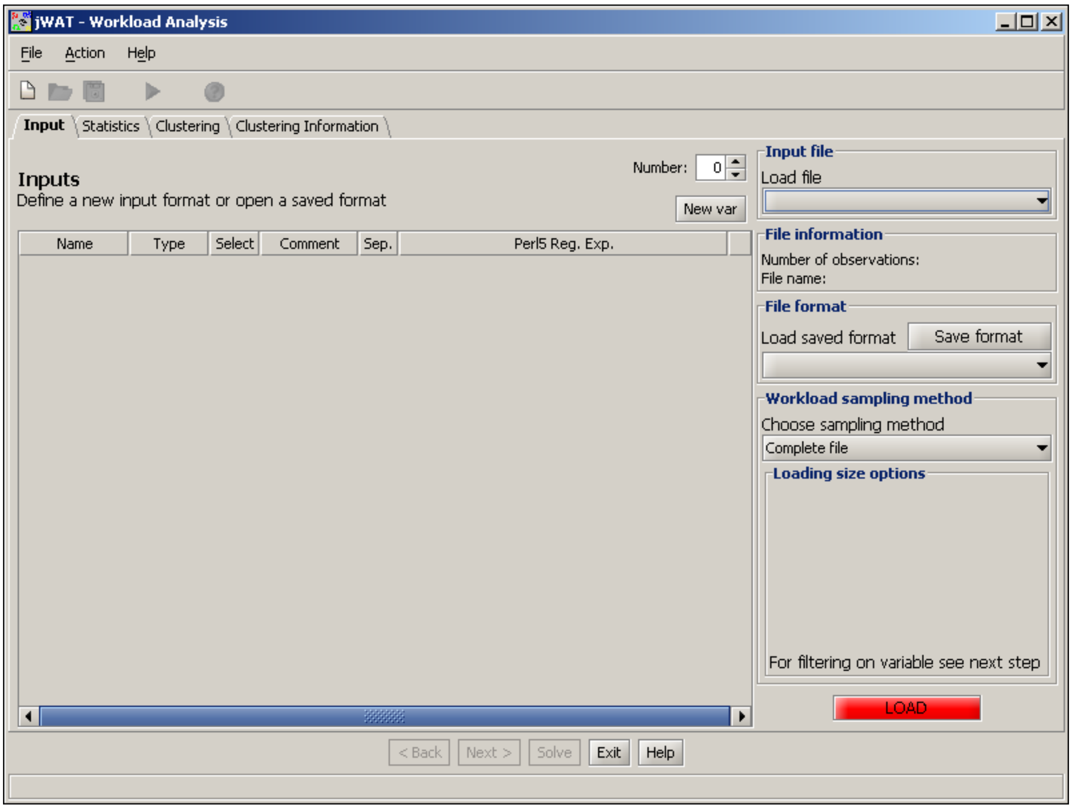
- *Men*: three groups of functions are available: File, Action, Help. To utilize the menu select the desired option. For the description of menu options see Section subsection 4.1.7.
- *Toolbar*: several buttons are available to facilitate and to speed up the access to the functions (e.g, New input file, Help ... see Section subsection 4.1.7). When the cursor is over a button tooltips appear.
- *Tabbed pane*: It concerns the main functions of the application. All the operations that can be performed on the input data are shown in four navigable tabs: input, statistics, clustering, clustering information (Figure 4.4).
- *Navigation buttons*: used for navigate between panels. The buttons are enabled and disabled automatically depending on the operations to be performed in the actual step (Figure 4.5).

In the following sections the utilization of the panels will be described in detail together with the available options.

4.1.1 A workload characterization study

In this section the main operations to be performed in a workload characterization study are described. The definition of the *workload basic components*, that is, the identification of the smallest level of detail that is to be considered and of the set of variables to be used for their description, is among the first operations to be performed. Depending on the intended use of the study, as workload basic components one may select interactive commands, database queries and updates, web site requests, web downloads, programs, scripts, servlets, applets, traffic intensity, source level instructions, etc. In the following we will refer to a workload basic component as an *observation* (i.e., an item in the input file) and to the parameters that describe each component as *variables*.

1. The first step consists of the selection of the file containing the data that are to be considered as input, of the variables to load and of their format (see Section 4.1.2). Let us remark that each observation may be described in the input file by n variables but that the number of variables that are considered in the actual statistical analysis may be less than n. It is also possible to decrease the number of observations



Inputs

Define a new input format or open a saved format

Number of observations:

File name:

For filtering on variable see next step

Figure 4.2: Main window for the Workload analysis

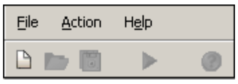


Figure 4.3: Men and application toolbar

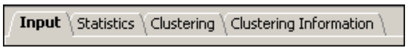


Figure 4.4: Tabs selector

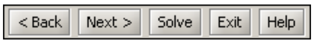


Figure 4.5: Navigation buttons

of the original input file (decrease its size) that will be loaded (considered) in the characterization study. Several filtering criteria for the construction of a sample of data are available: random selection of a given number of observations, selection of the observations that belong to a given interval of their sequence number. At the end of the data loading a panel describing the errors found in the data is shown. A user may decide to stop the analysis and verify the input data or may continue with the following steps.

2. Computation and visualization for each variable of the descriptive statistics univariate and bivariate, of the frequency graphs, of the scatter plots, and of the QQ-plot compared to a normal or exponential distribution (Section 4.1.3).
   It is important to observe that usually the variables are expressed in nonhomogeneous units. To make them comparable a *scale change* must be performed. The following statistical analysis of the original data, especially of the histograms of their parameters, allows us to determine when it would be useful to apply to them a logarithmic or another transformation. Very often the density functions of some variables are highly positively skewed and have high values of kurtosis too. Applying a logarithmic transformation of highly skewed variable densities it is possible to avoid or to reduce correlations among means and variances of the clusters that will be identified. Several *transformations* may be applied to the numeric variables (attention, *ONLY* to the numeric variables) .
   A *sample* may be extracted from input file by using different criteria. The distributions of two variables may be compared through QQ-plots and scatter plots selected in the corresponding matrix.

3. Selection of the clustering algorithm to apply and of the variables to be considered in the cluster analysis (see Section 4.1.4). The basic idea is to partition the given set of observations into classes of components having homogeneous characteristics. Among the various clustering algorithms the most commonly applied to workload characterization problems belong to the non-hierarchical k-means family. The algorithm produces a number of classes ranging from 1 to k, where k is a parameter that must be specified by the user. The observations are assigned to a class so that the sum of the distances between all the elements and the centroid of the class is minimized. A set of representative components will then be selected from the classes according to some extraction criteria (usually the centroids of the clusters) and used in the construction of the workload model.

4. A panel is available for the visualization of the results of the clustering analysis. Several graphics and tables are reported in order to describe the clusters composition and the goodness of a partition. Scatter plots are available for all the variables considered (see Section 4.1.5).

4.1.2 Input definition

The panel for the description of the input data (Figure 4.2) is a very important one since it allows the selection of the input file that contains the data to be analyzed and the description of their structure. In such a way we may take into consideration the log files generated by different systems and tools. The format definition structure of the input file is described in detail in Section section 4.2.

Input file selection

To select the input file use the panel shown in Figure 4.6. The *Browse...* option for the selection of the input file is available (Figure 4.6). Once a file has been selected its name is added to the list of the files available for following analyses. A panel showing the number of observations loaded and the file name will automatically appears (Figure 4.7).

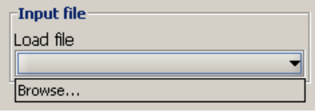
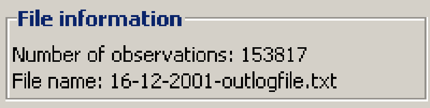


Figure 4.6: Panel for the selection of the input file


File information

Number of observations: 153817
File name: 16-12-2001-outlogfile.txt

Figure 4.7: Panel with information about the selected file

Format definition

The next operation consists of the definition of the data format. The format of the input data is required by the application:

- to recognize the number of variables that belongs to each observation
- to recognize the type (string, numeric or data) of each variable
- to the right interpretation of the meaning of each data element that belong to a single observation. For the definition of the elements regular expressions Perl5 (Section section 4.2) are used. The defined formats are stored in a library and can be browsed and reused.
- to select a subset of the variables of the observations that are in the original input file.

We will describe a very simple example of the format definition of an '*observation*, i.e., a single element in the input log file. The considered log file contains several data, e.g., IP client address that submits the request for a resource to a web server, request timestamp, etc...) that are called *variables*.

After the selection of the input file, the format definition table is empty, see Figure 4.2, since no information regarding current format has already been defined.

Two types of format definition are available:

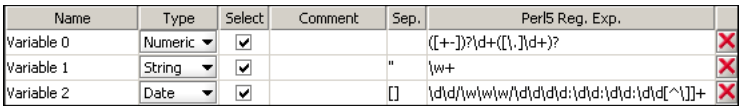
- Manual, that consists of the definition of the format of each single variable through relative table and relative commands
- Automatic, that load one of the predefined standard formats previously defined and loaded in files *'.jwat-format'*.

Manual definition

At the beginning, the correct number of variables of each observation in the log file is to be set (use the spinner *'Number'*). By using the '*New var*' button a new variable is added to the format table. The same result is obtained using the spinner *'Number'* at the right top of the table (see Figure 4.8).

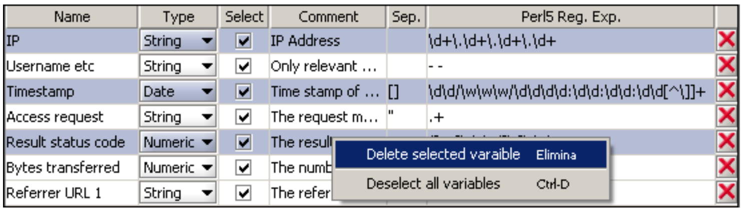
To remove one or more variables by a format table we can utilize:

- the button ☒ at the end of each row of the table. The result is the deletion of the concerning variable
- turned arrow towards the bottom of the spinner. The result is the deletion of the last variable of the table
- the right button of the mouse on a single selected variable, or on a set, to delete them from the format table (Figure 4.9).



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	
Variable 0	Numeric	☑			([+-])?\d+([\.]\d+)?	☒
Variable 1	String	☑		"	\w+	☒
Variable 2	Date	☑		[]	\d\d/\w\w\w/\d\d\d\d:\d\d:\d\d:\d\d[^\]]+	☒

Figure 4.8: Format table



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	
IP	String	☑	IP Address		\d+\.\d+\.\d+\.\d+	☒
Username etc	String	☑	Only relevant ...		- -	☒
Timestamp	Date	☑	Time stamp of ...	[]	\d\d/\w\w\w/\d\d\d\d:\d\d:\d\d:\d\d[^\]]+	☒
Access request	String	☑	The request m...	"	.+	☒
Result status code	Numeric	☑	The resul			☒
Bytes transferred	Numeric	☑	The numb			☒
Referrer URL 1	String	☑	The refer			☒

Delete selected varaible Elimina
Deselect all variables Ctrl-D

Figure 4.9: Deletion of selected variables

Once the correct number of variables in a observation is defined the next step is their description. The columns of the format table have the following meaning:

- *Name:* as default, to each new added variable it is assigned the standard name *Variable X.* In order to facilitate the analysis of the results it is recommended to change these names with new ones related with the meaning of the variables.

- *Type:* it identifies the type of the variable. Possible choices are *number, string* and *data*. The type definition is fundamental for the right interpretation of the results.
- *Selection:* it allows to specify which of the variables of the observations registered in the input file will be effectively loaded. In any case, it is necessary to define all the variables of each observation, also if some of them will not be selected for the statistical analysis.
- *Comment:* it is allowed to add a string of comment to each variable.
- *Delimiter [Sep]:* it identifies the character used (if it exists) in the input file to delimit the field identified by the current variable. For example in the Apache server log file the variable "timestamp" is contained within square brackets.
- *Regular Expression :* definition of regular expression (Perl 5) that defines the variable. For example in case of IP variables a possible regular expression is: ‘\d+\.\d+\.\d+\.\d+’.
- *Delete:* the corresponding variable will be deleted.

In order to help the users for each of the types the tool sets automatically some of the fields (delimiter of field and regular expression) supplying standard settings as shown in Figure 4.8 where for for the type *string* it is proposed " as delimiter and \w+ as regular expression.

A format definition may be saved by using the ‘*Save format*’ button so that it may reused in the future.

Format loading

If a standard log file (Apache, IIS) is used it is possible to load the corresponding format by using the combobox in the panel of Figure 4.10.

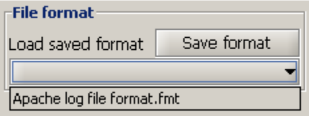
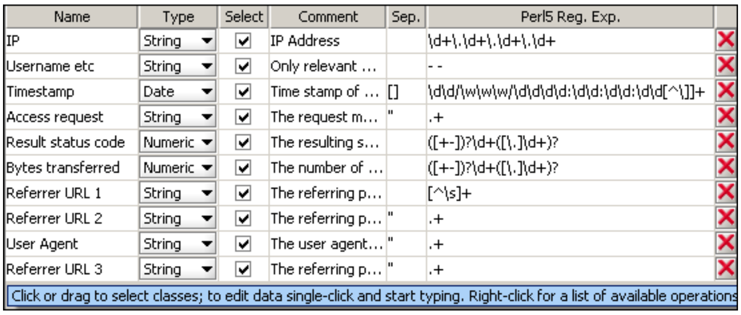


Figure 4.10: Panel for the loading of standard and previously saved format

For example, by selecting the option “Apache log file format”, the format definition table of Figure 4.11 is automatically loaded. At this point, the user can select through the format table selection field the subset of variables to load.



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	
IP	String	☑	IP Address		\d+\.\d+\.\d+\.\d+	☒
Username etc	String	☑	Only relevant ...		- -	☒
Timestamp	Date	☑	Time stamp of ...	[]	\d\d/\w\w\w/\d\d\d\d:\d\d:\d\d:\d\d[^\]]+	☒
Access request	String	☑	The request m...	"	.+	☒
Result status code	Numeric	☑	The resulting s...		([+-])?\d+([\.]\d+)?	☒
Bytes transferred	Numeric	☑	The number of ...		([+-])?\d+([\.]\d+)?	☒
Referrer URL 1	String	☑	The referring p...		[^\s]+	☒
Referrer URL 2	String	☑	The referring p...	"	.+	☒
User Agent	String	☑	The user agent...	"	.+	☒
Referrer URL 3	String	☑	The referring p...	"	.+	☒

Click or drag to select classes; to edit data single-click and start typing. Right-click for a list of available operations

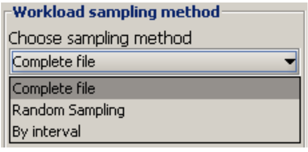
Figure 4.11: Format definition table

Methods for extracting a sample

In order to process the data from real log files, that usually are constituted by a very large number of observations (e.g., several Gigabytes), by a cluster algorithm with a reasonable amount of processing time and storage dimensions, the size of the original data set has to be reduced. Thus, only a sample drawn from the original set of data is often submitted as input to the clustering algorithm.

After the definition of the data format it is possible through the panel *Workload sampling method* of Figure 4.12 to select one of the following extraction criteria:

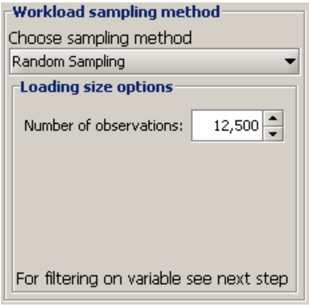
- *Complete file:* all the observations of the input file are considered.
- *Random:* the number of observations to be loaded should be specified in the panel of Fig.Figure 4.13.


Workload sampling method

Choose sampling method

Complete file

Figure 4.12: Extraction criteria for the sample construction


Workload sampling method

Choose sampling method

Random Sampling

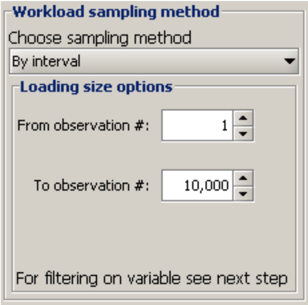
Loading size options

Number of observations: 12,500

For filtering on variable see next step

Figure 4.13: Random method for the construction of a sample

- *Interval:* the panel of Figure 4.14 that requires to specify the interval of observations (according to their id numbers) to load will be shown.


Workload sampling method

Choose sampling method

By interval

Loading size options

From observation #: 1

To observation #: 10,000

For filtering on variable see next step

Figure 4.14: Interval method options

Loading of log file

After the definition of the input file the loading of the selected observations will be started. After pressing the '*LOAD*' button (Figure 4.15) the loading window of Figure 4.16 will be opened. The percentage of loading completion, the number of processed observations and the number of observations discarded because not consistent with the specified format are shown. At the end of the load a window summarizing the total number of observations loaded and the total number of observations loaded correctly (Figure 4.17) is shown. It is possible to visualize the log file that contains the description of the errors found during the loading operation (Figure 4.18). The error messages that are contained in the log file are:

- *Error in row xx : Element y is wrong*: (if you defined separators) the element corresponding to the variable *y*, for the row *xx*, does not correspond to the defined regular expression.

- *Error in row xx : Line does not match format (element y not found)*: The element corresponding to the variable *y* is not present in the row (observation) *xx* of the input file.

- *Error in row xx : Too many fields* : the row *xx* contains elements in excess compared to the defined format. This may indicate with high probability that the format definition described by the user is not correct for the considered observation.

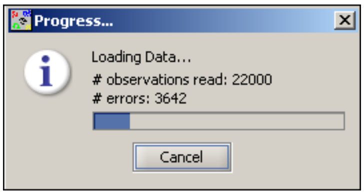
4.1.3 Data processing

A tabs substructure (Figure 4.19) shows the groups of the various statistics available:

Univariate: descriptive statistics, graphs and transformations.

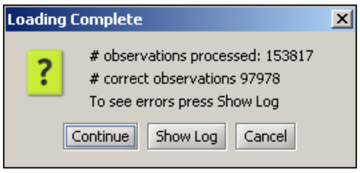
Sample construction: methods of sample construction and data filtering.

Figure 4.15: Button for the loading of the input data.



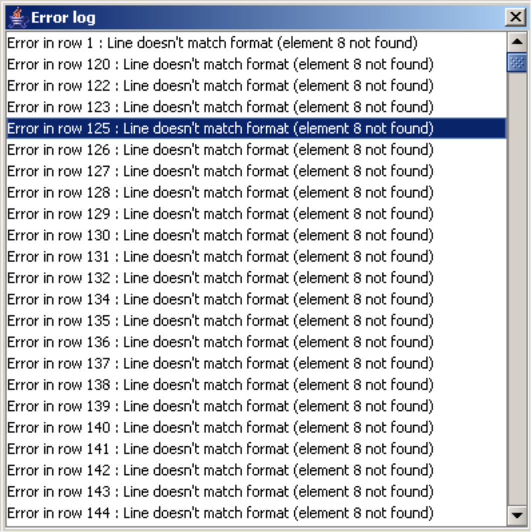
Loading Data...
# observations read: 22000
# errors: 3642

Figure 4.16: Window showing the loading progress



observations processed: 153817
correct observations 97978
To see errors press Show Log

Figure 4.17: Loading phase results



```
Error in row 1 : Line doesn't match format (element 8 not found)
Error in row 120 : Line doesn't match format (element 8 not found)
Error in row 122 : Line doesn't match format (element 8 not found)
Error in row 123 : Line doesn't match format (element 8 not found)
Error in row 125 : Line doesn't match format (element 8 not found)
Error in row 126 : Line doesn't match format (element 8 not found)
Error in row 127 : Line doesn't match format (element 8 not found)
Error in row 128 : Line doesn't match format (element 8 not found)
Error in row 129 : Line doesn't match format (element 8 not found)
Error in row 130 : Line doesn't match format (element 8 not found)
Error in row 131 : Line doesn't match format (element 8 not found)
Error in row 132 : Line doesn't match format (element 8 not found)
Error in row 134 : Line doesn't match format (element 8 not found)
Error in row 135 : Line doesn't match format (element 8 not found)
Error in row 136 : Line doesn't match format (element 8 not found)
Error in row 137 : Line doesn't match format (element 8 not found)
Error in row 138 : Line doesn't match format (element 8 not found)
Error in row 139 : Line doesn't match format (element 8 not found)
Error in row 140 : Line doesn't match format (element 8 not found)
Error in row 141 : Line doesn't match format (element 8 not found)
Error in row 142 : Line doesn't match format (element 8 not found)
Error in row 143 : Line doesn't match format (element 8 not found)
Error in row 144 : Line doesn't match format (element 8 not found)
```

Figure 4.18: Example of error log in the loading phase

Bivariate: bivariate statistics.

QQ-plot matrix: QQ-plots of the variables for comparison with normal and exponential distributions

Scatter plot matrix: Scatter plots of all the variables.

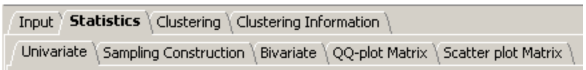
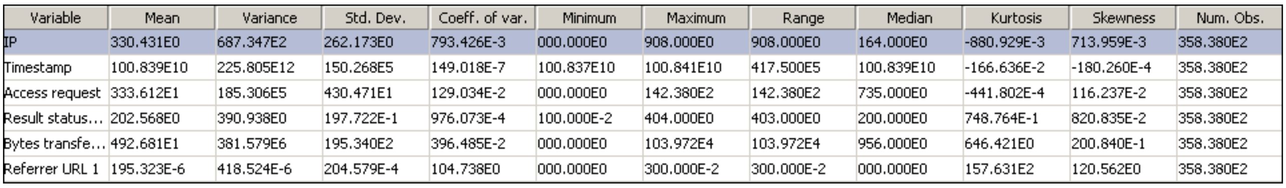


Figure 4.19: Tabs structure for the processing of the data

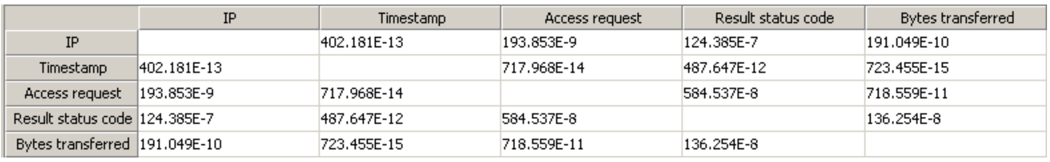
Statistics

The univariate and bivariate panels show respectively several statistical indexes concerning each variable of the observations (Figure 4.20) and the correlation coefficients of the variables (Figure 4.21).



Variable	Mean	Variance	Std. Dev.	Coeff. of var.	Minimum	Maximum	Range	Median	Kurtosis	Skewness	Num. Obs.
IP	330.431E0	687.347E2	262.173E0	793.426E-3	000.000E0	908.000E0	908.000E0	164.000E0	-880.929E-3	713.959E-3	358.380E2
Timestamp	100.839E10	225.805E12	150.268E5	149.018E-7	100.837E10	100.841E10	417.500E5	100.839E10	-166.636E-2	-180.260E-4	358.380E2
Access request	333.612E1	185.306E5	430.471E1	129.034E-2	000.000E0	142.380E2	142.380E2	735.000E0	-441.802E-4	116.237E-2	358.380E2
Result status...	202.568E0	390.938E0	197.722E-1	976.073E-4	100.000E-2	404.000E0	403.000E0	200.000E0	748.764E-1	820.835E-2	358.380E2
Bytes transfe...	492.681E1	381.579E6	195.340E2	396.485E-2	000.000E0	103.972E4	103.972E4	956.000E0	646.421E0	200.840E-1	358.380E2
Referrer URL 1	195.323E-6	418.524E-6	204.579E-4	104.738E0	000.000E0	300.000E-2	300.000E-2	000.000E0	157.631E2	120.562E0	358.380E2

Figure 4.20: Descriptive statistics of the variables



	IP	Timestamp	Access request	Result status code	Bytes transferred
IP		402.181E-13	193.853E-9	124.385E-7	191.049E-10
Timestamp	402.181E-13		717.968E-14	487.647E-12	723.455E-15
Access request	193.853E-9	717.968E-14		584.537E-8	718.559E-11
Result status code	124.385E-7	487.647E-12	584.537E-8		136.254E-8
Bytes transferred	191.049E-10	723.455E-15	718.559E-11	136.254E-8	

Figure 4.21: Correlation coefficients of the variables

The univariate panel shows also the following two types of graphs:

- a histogram, or frequency graph, preview for each variable, that can be magnified with a double click on the preview (Figure 4.22).
- a preview of the QQ-plot graph of the variable quantiles compared with the normal distribution quantiles with the same average and variance or the exponential distribution with the same average. Also this preview can be magnified with a double click (Figure 4.23).

Both magnified graphs can be saved in *eps* (Encapsulated PostScript) and *png* (Portable Network Graphics) formats.

Transformations and sample extraction

The panel of univariate statistics allows also to perform some variables transformations. To apply a transformation select the variable in the table of Figure 4.20; if it is of a numeric type the panel of Figure 4.24 that allows the selection of the type of transformation (logarithmic, min-max and standard deviation) will be shown and press the '*Apply transformation*' button. It is possible to undo the last transformation by pressing the '*Undo transformation*' button. The list of the variables with the transformations applied is reported in the table. If the selected variable is not numeric, the following message is shown: *"This variable could not be transformed since it is not numeric"*.

To extract a sample from the original input file press on tab *'Sample construction'*. The panel of Figure 4.25 shows the possible criteria that can be applied to the loaded variables that depends on the selected method and on the type of the variables.

- *Random* and *Obsev. # interval* methods are similar to those available in input window and allow to select randomly n observations from the input file or to extract the observations whose numerical identifiers are included into the interval defined by the user.

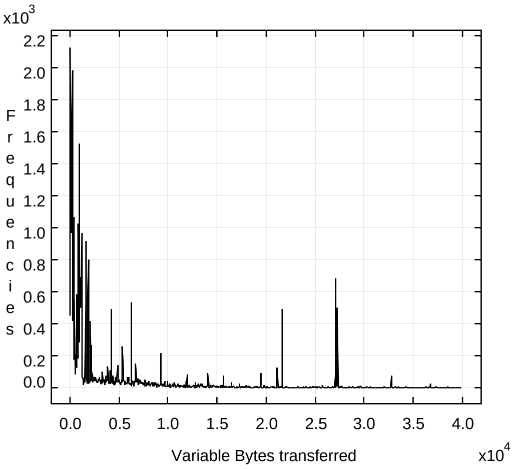


Figure 4.22: Histogram, or frequency graph, of the variable "Bytes transferred"

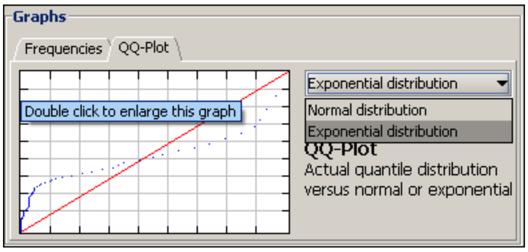
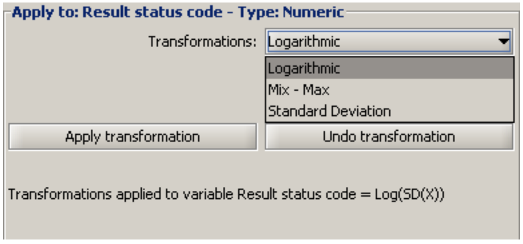


Figure 4.23: QQ-plot preview for the comparison of a given distribution with an exponential or a normal ones.


Apply to: Result status code - Type: Numeric

Transformations: Logarithmic

Transformations applied to variable Result status code = Log(SD(X))

Figure 4.24: Selection of the transformation to apply to the original data

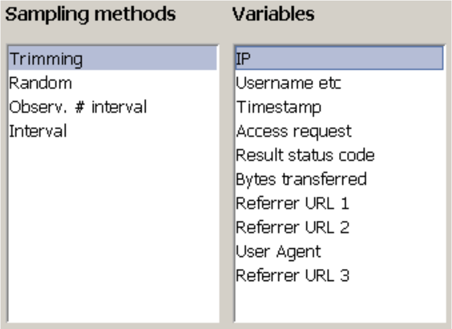
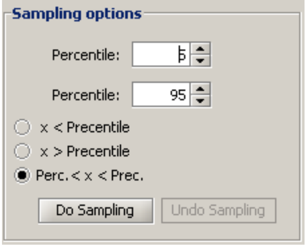


Figure 4.25: Sampling extraction criteria that can be applied on the selected variables

- *Trimming :* the outliers, i.e., those values of a variable that are too distant from the other values of the same variable, may distort the transformation and the statistical analysis by causing the other observations to be assigned too much or too little weight should be removed. To filter a variable distribution (trimming operation) the percentiles that should be removed can be specified (Figure 4.26).



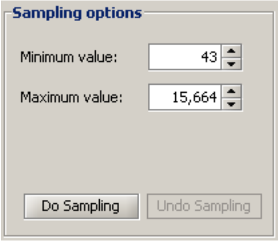
Percentile: 5

Percentile: 95

○ x < Precentile
○ x > Precentile
● Perc. < x < Prec.

Figure 4.26: Filter of the percentiles of a variable (trimming of the distribution)

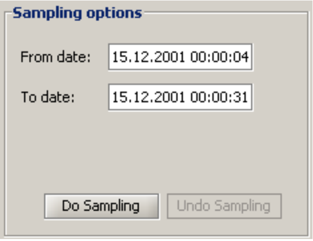
- *Interval* operation applies a filter on a variable's values. Depending on the type of the variable different options panels will be shown. If the selected variable is *numeric* the panel is similar to that of Figure 4.27: the minimum and maximum values of the variable should be specified. If the selected variable is of *data* type the panel is similar to that of Figure 4.28: the dates of start and end of observations should be specified. If the variable is of *string* type, it is possible to specify a substring that has to be contained in the selected observations.



Minimum value: 43

Maximum value: 15,664

Figure 4.27: Filter on the values of a numeric variable



From date: 15.12.2001 00:00:04

To date: 15.12.2001 00:00:31

Figure 4.28: Filter on the values of a data variable

The operation of *'Undo sampling'* is available.

Graphs

The last two panels of the statistics tab concern a preview of QQ-plot graphs and of scatter plots for all the variables, respectively. Both allow to save graphs in the most common formats(*.png, .eps*). Previews can be magnified with a left double click of the mouse. Several functions are available for the graphs: portions zoom, points dimension, by clicking the right button of the mouse on a graph various formats are available for export the graph as image.

4.1.4 Clustering algorithms

The next step of the characterization process is the choice of the clustering algorithm to be used ad of the options available. Two clustering algorithms are actually implemented in the tool (k-Means and Fuzzy k-Means), see Figure 4.31. Depending on the selected algorithm, in the right panel of tab the available options are listed.

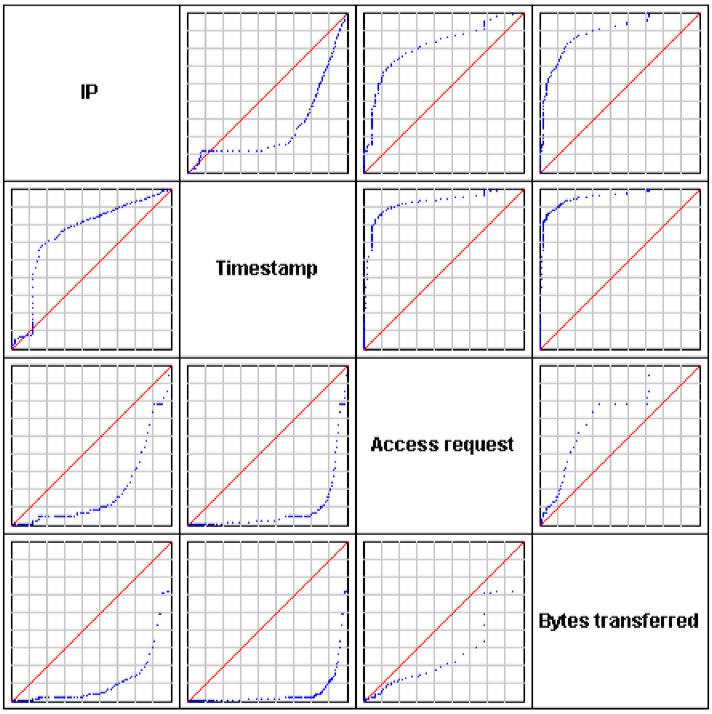


Figure 4.29: QQ-plot matrix for the comparison of the distributions of two variables

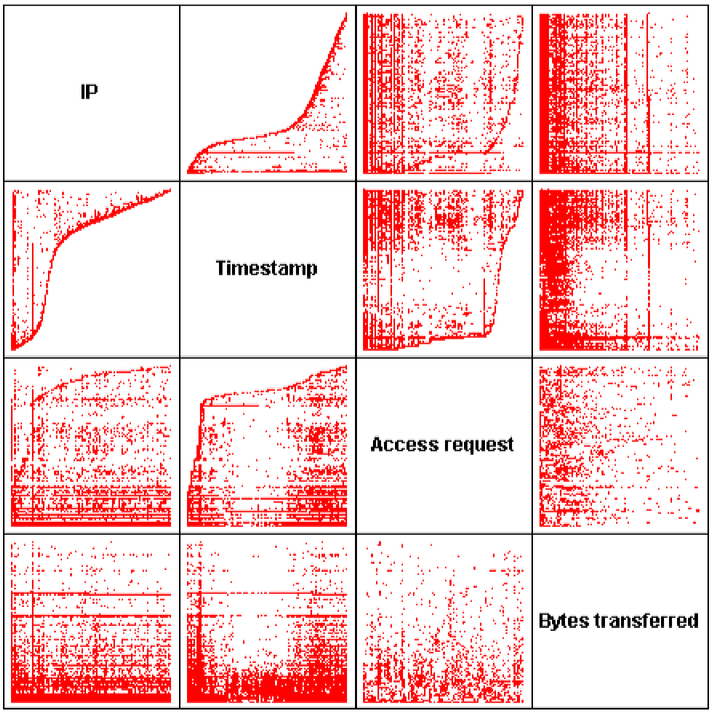
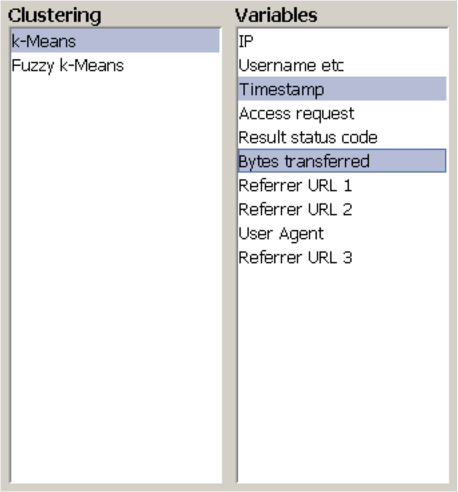


Figure 4.30: Scatter matrix

The execution of a clustering algorithm will start when the buttons '*Solve*' or ▶ are pressed. During the execution, the window of Figure 4.32 that report the progress state and allows to interrupt the execution is shown.



Clustering

k-Means

Fuzzy k-Means

Variables

IP

Username etc

Timestamp

Access request

Result status code

Bytes transferred

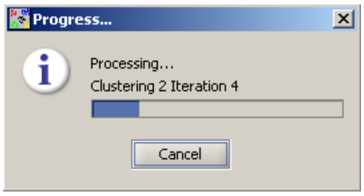
Referrer URL 1

Referrer URL 2

User Agent

Referrer URL 3

Figure 4.31: Clustering algorithm used and variables selected that will be considered in the analysis.



Progress...

Processing...

Clustering 2 Iteration 4

Figure 4.32: Clustering progress panel

K-Means algorithm

The non-hierarchical k-Means algorithm implemented in the tool requires the specification of the following parameters, Figure 4.33:

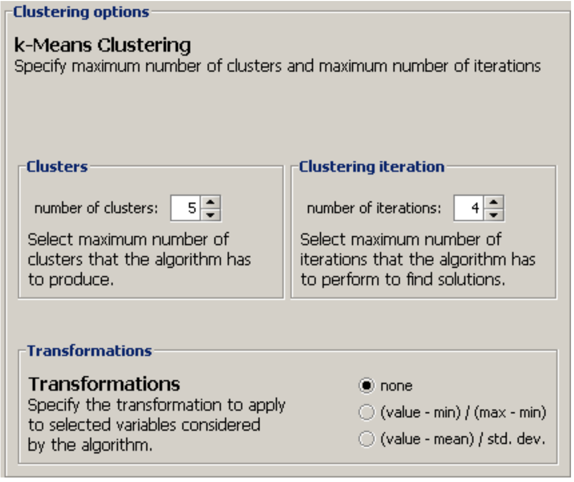
- *the maximum number k of clusters* that the algorithm has to produce. Note that the algorithm produces the results for all partitions from 1 to k clusters

- *the maximum number of iterations* to perform in order to find the optimal partition. The initial subdivision into k clusters is iteratively improved by shifting, based on the selected criterion (in this case the Euclidean distance), the elements of a cluster to another and computing after each assignment the new center of mass of the clusters. The optimum configuration is obtained when points can no longer be reassigned. The selected value is an upper limit of the number of interactions for each partition. Experiences suggest the value of 4 interactions as a reasonable choice: the results are obtained in a short time and are enough accurate. To obtain more accurate results higher values should be used, this involves a higher computation time

- *the transformation type* to apply to selected variables (if needed). Transformations of the values are applied before the execution of the algorithm and at the end of the execution the results can be transformed back to their original values. The transformation of the values is often required since the variables are usually expressed in different units and their ranges are very different. Since the algorithm uses the Euclidean distance function as comparison metric to determine if an observation belongs to a cluster, the results could be not reliable if the values differ of one or more order of magnitude.

Fuzzy k-Means algorithm

The fuzzy k-Means algorithm implemented in the tool requires the following parameters, Figure 4.34:



Clustering options

k-Means Clustering

Specify maximum number of clusters and maximum number of iterations

Clusters

number of clusters: 5

Select maximum number of clusters that the algorithm has to produce.

Clustering iteration

number of iterations: 4

Select maximum number of iterations that the algorithm has to perform to find solutions.

Transformations

Transformations

Specify the transformation to apply to selected variables considered by the algorithm.

● none

○ (value - min) / (max - min)

○ (value - mean) / std. dev.

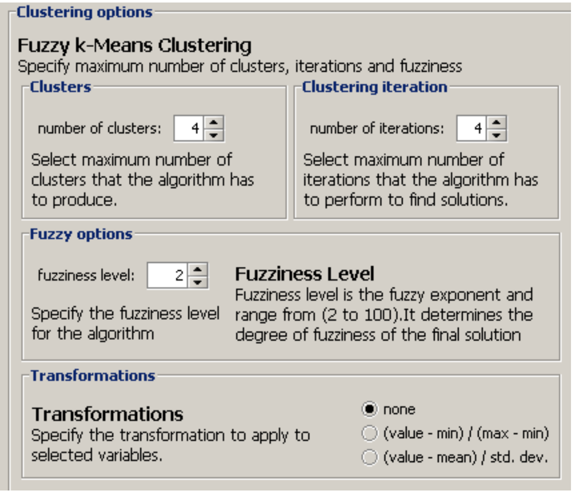
Figure 4.33: Parameters of the clustering algorithm k-Means

- *the maximum number k of clusters* that the algorithm has to produce. The results for all the partitions from two to the selected maximum value will be provided. The higher is this value the higher is the execution time of the algorithm

- *the maximum number of iterations* to perform in order to find the optimal partition. Four is a suggested value. See the comments reported above for the K-means algorithm.

- *the fuzziness level* to apply during the algorithm execution. It is the value used by algorithm to determine the fuzziness degree of final solution. It ranges from 2 to 100

- *the transformation type* to apply to selected variables (if needed). See the comments reported above for the K-means algorithm.



Clustering options

Fuzzy k-Means Clustering

Specify maximum number of clusters, iterations and fuzziness

Clusters

number of clusters: 4

Select maximum number of clusters that the algorithm has to produce.

Clustering iteration

number of iterations: 4

Select maximum number of iterations that the algorithm has to perform to find solutions.

Fuzzy options

fuzziness level: 2

Specify the fuzziness level for the algorithm

Fuzziness Level

Fuzziness level is the fuzzy exponent and range from (2 to 100).It determines the degree of fuzziness of the final solution

Transformations

Transformations

Specify the transformation to apply to selected variables.

● none

○ (value - min) / (max - min)

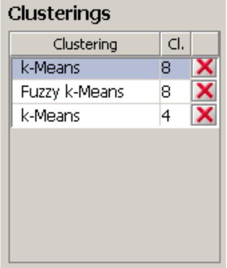
○ (value - mean) / std. dev.

Figure 4.34: Parameters of the clustering algorithm Fuzzy k-Means

4.1.5 The results of a clustering execution

At the end of the execution of a clustering algorithm the tab that allows the analysis of the results is shown. The main panel shows the table in Figure 4.35 that contains the algorithm(s) executed and the maximum number of clusters identified. It is possible to delete the results of an execution by clicking the button ☒ in the last column.

Depending on the selected clustering algorithm different results panels are shown in the results table.



Clusterings

Clustering	Cl.	
k-Means	8	☒
Fuzzy k-Means	8	☒
k-Means	4	☒

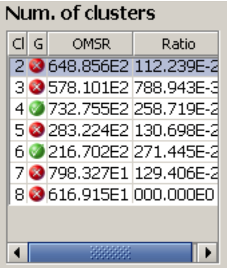
Figure 4.35: List of executed clustering algorithms

K-Means algorithm results

When the $k - Means$ algorithm results are selected, a table reporting the indices concerning the goodness of the partitions is shown, Figure 4.36.
To estimate the optimal number k of clusters in which the given set of observations will be subdivided two indicators of the goodness of a partition are used. For each variable the *Overall Mean Square Ratio, OMSR*, i.e., a measure of the reduction of within-cluster variance between partitions in k and in $k + 1$ clusters, and the *ratio* of the variance among the clusters and the within-cluster variance have been used. Large values of the $OMSR$ justify increasing the number of clusters from k to $k + 1$. An optimal partition should also be characterized by values greater than 1 of the *ratio*s of the variances among the clusters and within the clusters for each variable.

A visual indication of the goodness of a partition is given with the icons ✓ and ✗ representing good and bad results respectively. The values of OMSR and of the *ratio* are also reported in the table.

By selecting one of the rows of the table in Figure 4.36 the panel is updated and in the right side a tabs structure concerning the visualization of the results is shown. Three main panels are available (Figure 4.37):



Num. of clusters

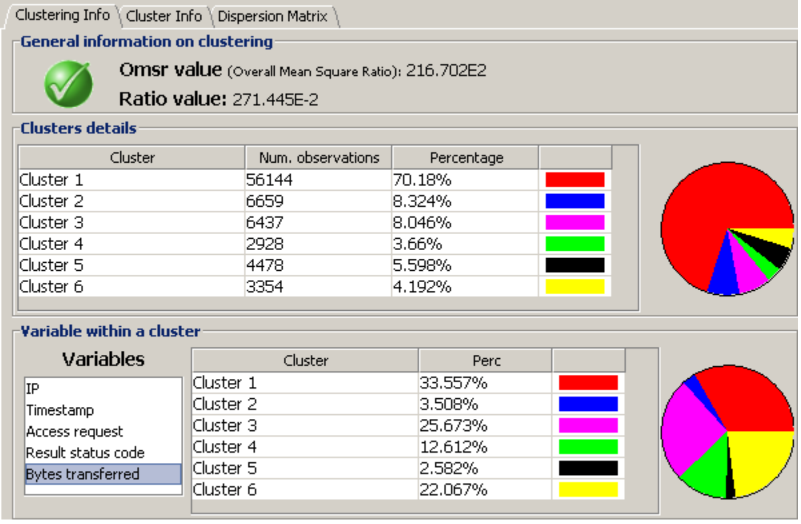
Cl	G	OMSR	Ratio
2	✗	648.856E2	112.239E-2
3	✗	578.101E2	788.943E-3
4	✓	732.755E2	258.719E-2
5	✗	283.224E2	130.698E-2
6	✓	216.702E2	271.445E-2
7	✗	798.327E1	129.406E-2
8	✗	616.915E1	000.000E0

Figure 4.36: Indicators of the goodness of a partition (*Overall Mean Square Ratio* and *ratio* indices) for the K-Means algorithm

- *Clustering Info:* It shows some statistical information concerning the clusters. For each cluster the number of its components and its weight with respect to the total population are reported. Pie-charts are also used to visualize these data. The panel at the bottom shows for each variable the distribution (in %) of its values among the clusters.

- *Clusters Info:* It shows statistics concerning the clusters (Figure 4.38). Besides the classic descriptive univariate statistics, the *ISC*, that indicates which variables have been used for the clustering, and the *center* values, that refer to the centroid coordinates of each cluster, are reported. The panel at the bottom shows a preview (it can be magnified by a left double click of the mouse) of the graph concerning the comparisons of the distributions of the variables used in each cluster. It is also possible to visualize the list of observations that belong to a cluster through the *'Show Observations'* button (attention! this operation may requires several minutes).

- *Dispersion Matrix:* this panel shows the matrix of all the possible variable *vs* variable scatter plots (Figure 4.39). These graphs allows a visual interpretation of the compositions of the clusters since the observations are represented with the color corresponding to the cluster it is assigned. With a left double click of the mouse on a preview, the graph is magnified and several options can be selected with a right click on the graph.



General information on clustering

Omsr value (Overall Mean Square Ratio): 216.702E2

Ratio value: 271.445E-2

Clusters details

Cluster	Num. observations	Percentage	
Cluster 1	56144	70.18%	🟥
Cluster 2	6659	8.324%	🟦
Cluster 3	6437	8.046%	🟪
Cluster 4	2928	3.66%	🟩
Cluster 5	4478	5.598%	⬛
Cluster 6	3354	4.192%	🟨

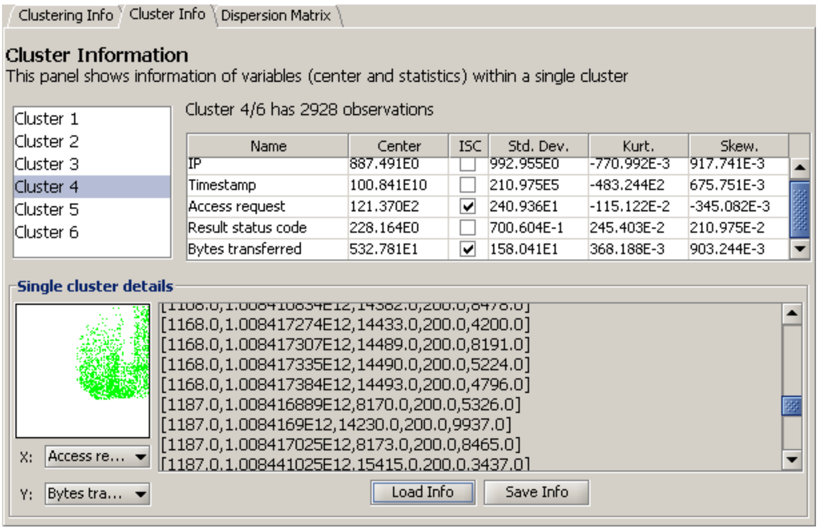
Variable within a cluster

Variables

- IP
- Timestamp
- Access request
- Result status code
- Bytes transferred

Cluster	Perc	
Cluster 1	33.557%	🟥
Cluster 2	3.508%	🟦
Cluster 3	25.673%	🟪
Cluster 4	12.612%	🟩
Cluster 5	2.582%	⬛
Cluster 6	22.067%	🟨

Figure 4.37: Characterization of the clusters



Cluster Information

This panel shows information of variables (center and statistics) within a single cluster

- Cluster 1
- Cluster 2
- Cluster 3
- Cluster 4
- Cluster 5
- Cluster 6

Cluster 4/6 has 2928 observations

Name	Center	ISC	Std. Dev.	Kurt.	Skew.
IP	887.491E0	□	992.955E0	-770.992E-3	917.741E-3
Timestamp	100.841E10	□	210.975E5	-483.244E2	675.751E-3
Access request	121.370E2	☑	240.936E1	-115.122E-2	-345.082E-3
Result status code	228.164E0	□	700.604E-1	245.403E-2	210.975E-2
Bytes transferred	532.781E1	☑	158.041E1	368.188E-3	903.244E-3

Single cluster details

```
[1168.0,1.008410834E12,14382.0,200.0,8478.0]
[1168.0,1.008417274E12,14433.0,200.0,4200.0]
[1168.0,1.008417307E12,14489.0,200.0,8191.0]
[1168.0,1.008417335E12,14490.0,200.0,5224.0]
[1168.0,1.008417384E12,14493.0,200.0,4796.0]
[1187.0,1.008416889E12,8170.0,200.0,5326.0]
[1187.0,1.0084169E12,14230.0,200.0,9937.0]
[1187.0,1.008417025E12,8173.0,200.0,8465.0]
[1187.0,1.008441025E12,15415.0,200.0,3437.0]
```

X: Access re...

Y: Bytes tra...

Figure 4.38: Descriptive statistics of the clusters obtained with the k-means algorithm and their visual characterization through the dispersion matrix

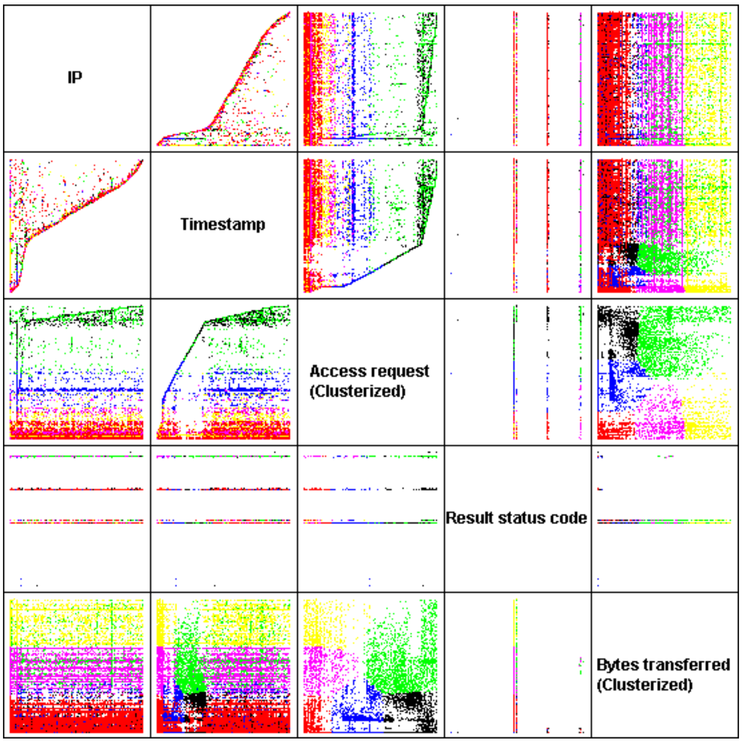
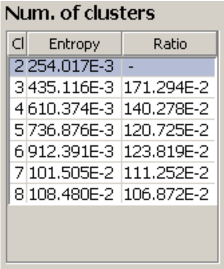


Figure 4.39: K-Means scatter plots matrix. Each observation is represented with the color corresponding to the cluster it is assigned.

Fuzzy K-Means algorithm results

When the fuzzy k-Means results are selected, the panel is uploaded and in the table of Figure 4.35 a new table containing a row for each partition identified is shown (Figure 4.36). The number of clusters, the *Entropy* index and the *ratio* are shown. By selecting one of the rows of table in Figure 4.40 the panel is updated and appears in the right side a tabs structure that permits to visualize information concerning results, by grouping them in three main panels:



Num. of clusters

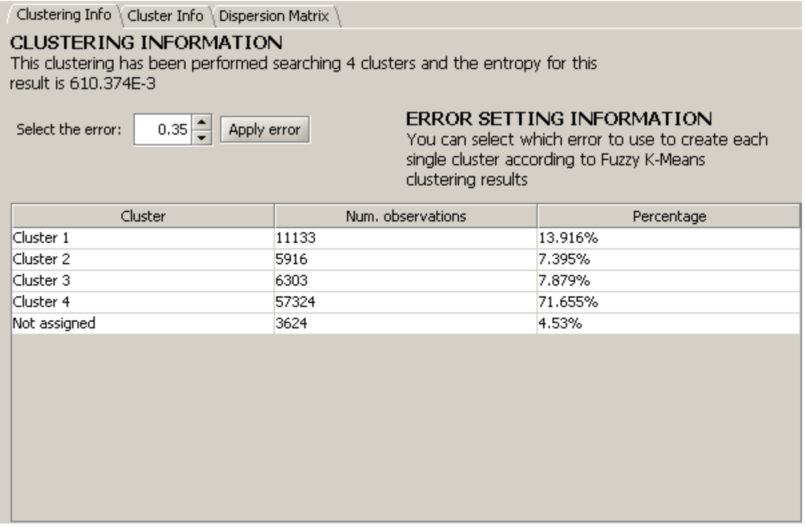
Cl	Entropy	Ratio
2	254.017E-3	-
3	435.116E-3	171.294E-2
4	610.374E-3	140.278E-2
5	736.876E-3	120.725E-2
6	912.391E-3	123.819E-2
7	101.505E-2	111.252E-2
8	108.480E-2	106.872E-2

Figure 4.40: Results of a fuzzy k-Means algorithm execution

- *Clustering Info:* It shows the clusters that have been identified and the entropy values (Figure 4.41). The error value used to identify a partition should be set by the user and can be easily changed (error setting information are reported on the right). Pressing the '*Apply error*' button the assignment of the observations in the current partitions will be changed according to the new error value and the table at the bottom showing the composition of each cluster (and the statistical and graphic information) of the following panels will be updated. Note that the sum of the percentages could be *greater* than 100% because of the multiple assignments of the fuzzy algorithm.

- *Clusters Info:* It shows statistical information concerning the selected cluster (Figure 4.42). See the comments reported above for the cluster info of the k-means algorithm.

- *Dispersion Matrix:* this last panel shows the matrix of all the possible variable vs. variable scatter plots (Figure 4.43). See the comments reported above for the dispersion matrix of the k-means algorithm.



CLUSTERING INFORMATION

This clustering has been performed searching 4 clusters and the entropy for this result is 610.374E-3

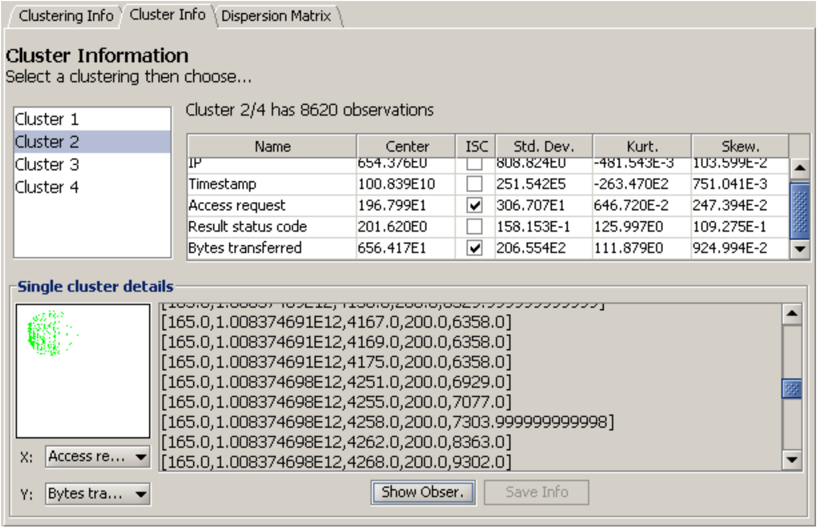
Select the error: 0.35 Apply error

ERROR SETTING INFORMATION

You can select which error to use to create each single cluster according to Fuzzy K-Means clustering results

Cluster	Num. observations	Percentage
Cluster 1	11133	13.916%
Cluster 2	5916	7.395%
Cluster 3	6303	7.879%
Cluster 4	57324	71.655%
Not assigned	3624	4.53%

Figure 4.41: Results of a fuzzy k-means algorithm execution



Cluster Information

Select a clustering then choose...

Cluster 1
Cluster 2
Cluster 3
Cluster 4

Cluster 2/4 has 8620 observations

Name	Center	ISC	Std. Dev.	Kurt.	Skew.
IP	654.376E0	□	808.824E0	-481.543E-3	103.599E-2
Timestamp	100.839E10	□	251.542E5	-263.470E2	751.041E-3
Access request	196.799E1	☑	306.707E1	646.720E-2	247.394E-2
Result status code	201.620E0	□	158.153E-1	125.997E0	109.275E-1
Bytes transferred	656.417E1	☑	206.554E2	111.879E0	924.994E-2

Single cluster details

```
[165.0,1.008374691E12,4167.0,200.0,6358.0]
[165.0,1.008374691E12,4169.0,200.0,6358.0]
[165.0,1.008374691E12,4175.0,200.0,6358.0]
[165.0,1.008374698E12,4251.0,200.0,6929.0]
[165.0,1.008374698E12,4255.0,200.0,7077.0]
[165.0,1.008374698E12,4258.0,200.0,7303.999999999998]
[165.0,1.008374698E12,4262.0,200.0,8363.0]
[165.0,1.008374698E12,4268.0,200.0,9302.0]
```

X: Access re...

Y: Bytes tra...

Figure 4.42: Descriptive statistics of the clusters obtained with the fuzzy k-means algorithm and their visual characterization through the dispersion matrix

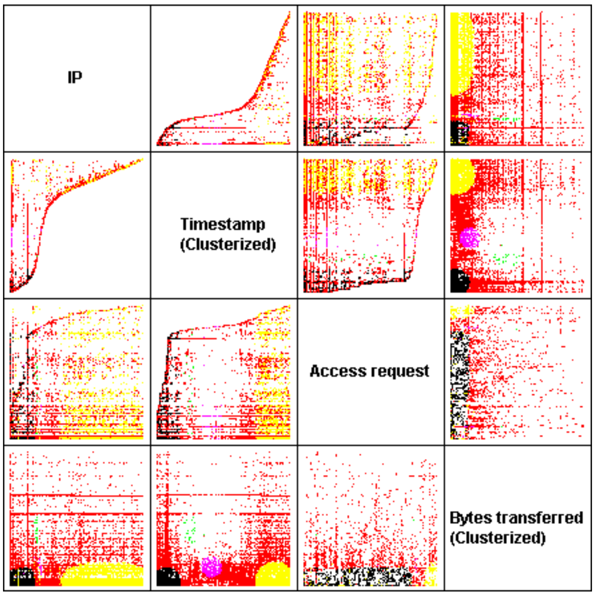
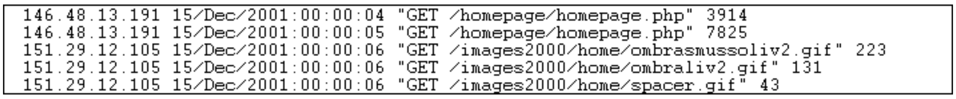


Figure 4.43: Fuzzy k-Means scatter plot matrix. Each observation is represented with the color corresponding to the cluster it is assigned

4.1.6 An example of a web log analysis

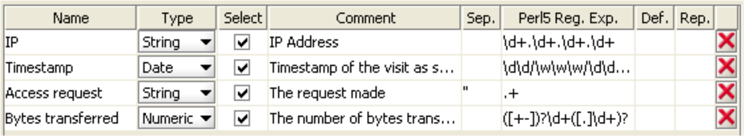
In this section we will describe an example of application of the JWAT tool for a workload analysis. The analyzed set of data consists of observations stored in the log file of a web server of a university site.
The input file *Demo1Data.jwat* and its format *Demo1Format.jwatformat* can be found in the subfolders *Examples* and *jwatFormats*, respectively, of the Java Modelling Tools folder installed with the JMT tools.

Step 1 - Input panel



```
146.48.13.191 15/Dec/2001:00:00:04 "GET /homepage/homepage.php" 3914
146.48.13.191 15/Dec/2001:00:00:05 "GET /homepage/homepage.php" 7825
151.29.12.105 15/Dec/2001:00:00:06 "GET /images2000/home/ombrasmussoliv2.gif" 223
151.29.12.105 15/Dec/2001:00:00:06 "GET /images2000/home/ombraliv2.gif" 131
151.29.12.105 15/Dec/2001:00:00:06 "GET /images2000/home/spacer.gif" 43
```

Figure 4.44: Example of observations in the input file considered.



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	Def.	Rep.	
IP	String	☑	IP Address		\d+.\d+.\d+.\d+			☒
Timestamp	Date	☑	Timestamp of the visit as s...		\d\d/\w\w\w/\d\d...			☒
Access request	String	☑	The request made	"	.+			☒
Bytes transferred	Numeric	☑	The number of bytes trans...		([+-])?\d+([.]\d+)?			☒

Figure 4.45: Input format example

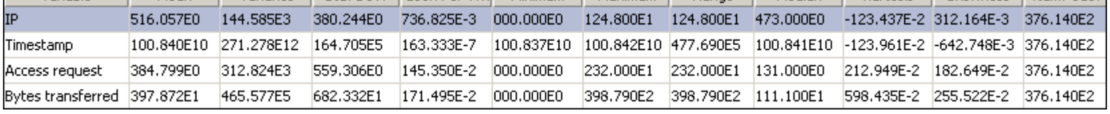
Each observation consists of the values of 4 variables described in Figure 4.45:

- *IP*: String type variable that defines IP address of the request.
- *Timestamp*: Date type variable that defines the timestamp of the request, the regular expression

  `"\d\d/\w\w\w/\d\d\d\d:\d\d:\d\d:\d\d"'`

  identifies that the data will be in the format dd/mmm/yyyy:hh:mm:ss (e.g., 12/Jun/2000:12:21:45).
- *Access request*: String type variable that identifies the object that is the target of the request. The definition through delimiter " and regular expression "'.*"' allows the reading of all the string that is contained between quotation marks; if it is necessary it is possible to modify the expression "'.*"' to read only a part of the string between quotation marks.
- *Bytes transferred*: Numeric type variable that identifies the quantity of bytes transferred due to the execution of the request submitted.

Step 2 - Descriptive statistics and sample extraction
Before the execution of the cluster analysis the size of the input data set has been reduced executing the filter *Interval* to the values of the variable *Bytes transferred* setting a maximum value of 20000 and a minimum value of 0. The purpose of this filtering operation is to obtain a data set more compact but still representative of the original one. In Tables Figure 4.46 and Figure 4.47 the descriptive statistics before and after applying the filter on the input data are shown, respectively. Scatter plot matrix of the filtered data are shown in Figure 4.48.

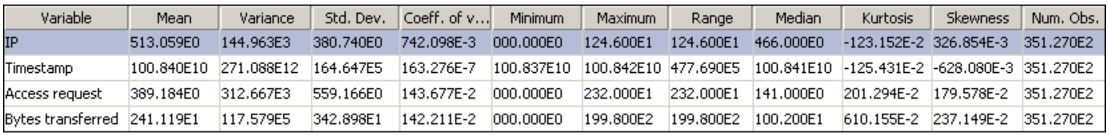


Variable	Mean	Variance	Std. Dev.	Coeff. of v...	Minimum	Maximum	Range	Median	Kurtosis	Skewness	Num. Obs.
IP	516.057E0	144.585E3	380.244E0	736.825E-3	000.000E0	124.800E1	124.800E1	473.000E0	-123.437E-2	312.164E-3	376.140E2
Timestamp	100.840E10	271.278E12	164.705E5	163.333E-7	100.837E10	100.842E10	477.690E5	100.841E10	-123.961E-2	-642.748E-3	376.140E2
Access request	384.799E0	312.824E3	559.306E0	145.350E-2	000.000E0	232.000E1	232.000E1	131.000E0	212.949E-2	182.649E-2	376.140E2
Bytes transferred	397.872E1	465.577E5	682.332E1	171.495E-2	000.000E0	398.790E2	398.790E2	111.100E1	598.435E-2	255.522E-2	376.140E2

Figure 4.46: Descriptive statistics of the original data before applying the filter on the "Bytes transferred" variable

About 2500 observations have been dropped due to the filtering action, reducing considerably the variance.

Step 3 - Clustering panel
The k-Means algorithm and the variables *timestamp* and *Bytes transferred* have been selected in the clustering panel together with the following options: number of clusters 7, interactions 50 and transformation *Max-Min*. The clustering results panel is opened automatically. Through the *'Back'* button we went back to the previous panel to execute a new clustering algorithm. The fuzzy k-Means algorithm and the variables *timestamp* and



Variable	Mean	Variance	Std. Dev.	Coeff. of v...	Minimum	Maximum	Range	Median	Kurtosis	Skewness	Num. Obs.
IP	513.059E0	144.963E3	380.740E0	742.098E-3	000.000E0	124.600E1	124.600E1	466.000E0	-123.152E-2	326.854E-3	351.270E2
Timestamp	100.840E10	271.088E12	164.647E5	163.276E-7	100.837E10	100.842E10	477.690E5	100.841E10	-125.431E-2	-628.080E-3	351.270E2
Access request	389.184E0	312.667E3	559.166E0	143.677E-2	000.000E0	232.000E1	232.000E1	141.000E0	201.294E-2	179.578E-2	351.270E2
Bytes transferred	241.119E1	117.579E5	342.898E1	142.211E-2	000.000E0	199.800E2	199.800E2	100.200E1	610.155E-2	237.149E-2	351.270E2

Figure 4.47: Descriptive statistics of the original data after applying the filter on the "Bytes transferred" variable

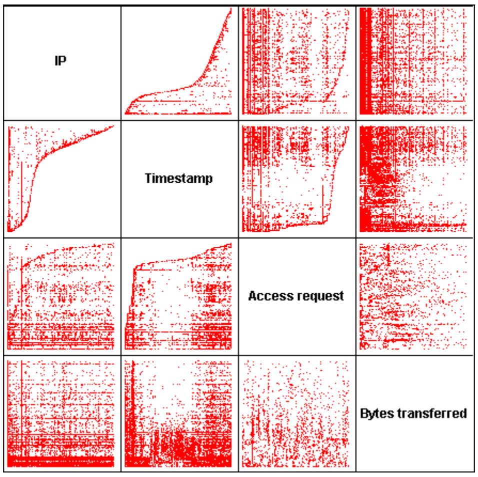
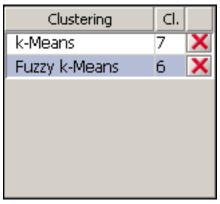


Figure 4.48: Scatters matrix on the data set after applying the filter on the "Bytes transferred" variable.

Bytes transferred have been selected together with the following options: number of clusters 6, interactions 20, fuzziness level 2.0.

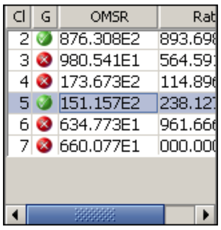
Step 4 - Results panel

The results concerning the goodness of the partitions obtained with the two clustering algorithms are reported in the following tables:



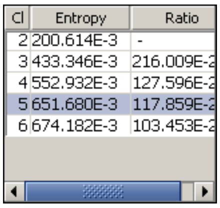
Clustering	Cl.	
k-Means	7	☒
Fuzzy k-Means	6	☒

Clustering algorithms



Cl	G	OMSR	Rat
2	✓	876.308E2	893.69
3	☒	980.541E1	564.59
4	☒	173.673E2	114.89
5	✓	151.157E2	238.12
6	☒	634.773E1	961.66
7	☒	660.077E1	000.00

k-Means



Cl	Entropy	Ratio
2	200.614E-3	-
3	433.346E-3	216.009E-
4	552.932E-3	127.596E-
5	651.680E-3	117.859E-
6	674.182E-3	103.453E-

Fuzzy k-Means

We have visualized different statistics for each algorithm by varying the number of clusters, in particular we show the scatter plots matrix for the partition with 5 clusters (as can be seen from the OMSR values this partition has been evaluated as a good one).

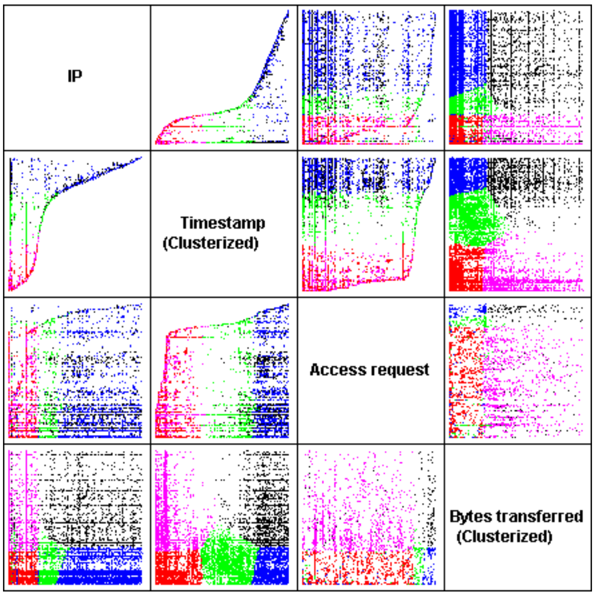


Figure 4.49: K-Means algorithm scatter plots matrix

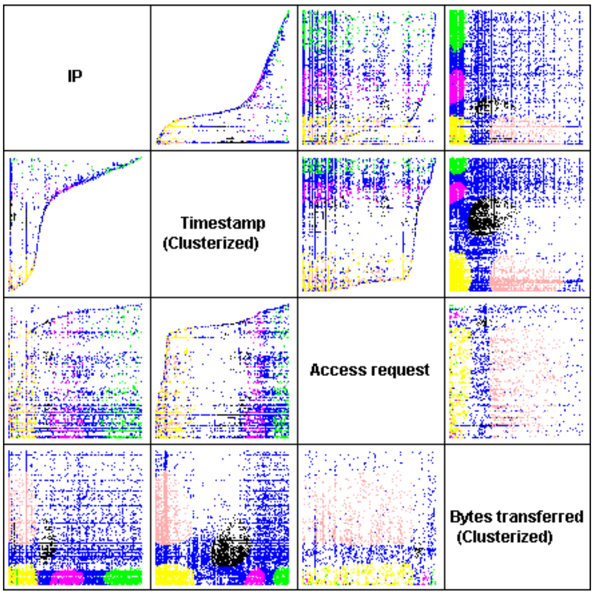


Figure 4.50: Fuzzy k-Means algorithm scatter plots matrix

4.1.7 Menus description

File

New Use this command to start a new analysis selecting a new log file.

Shortcut on Toolbar:	🗋
Accelerator Key:	CTRL+N

Exit

Use this command to terminate the JWAT current session. It is possible to utilize '*Exit*' button in the buttons bar of application. If the session has been modified after its creation or after its last saving, a popup window asks confirmation to save session.

Accelerator Key:	CTRL+Q

Action

Solve Use this command, when enabled, to start the algorithm selected in the clustering panel. It is possible to obtain the same result by using '*Solve*' button in the buttons bar of the application.

Shortcut on Toolbar:	▶
Accelerator Key:	CTRL+L

Help

JWAT Help Use this command to visualize the help. Suggestions concerning the single tab in the application are also available by using the '*Help*' button in the buttons bar of application.

Shortcut on Toolbar:	?
Accelerator Key:	CTRL+Q

About

Use this command to visualize information concerning JWAT application.

4.2 Format definition

In this section we will describe the format definition of the file in input to the application. We will introduce the basic concepts on which the format specific is based and then we will describe some examples with different files.

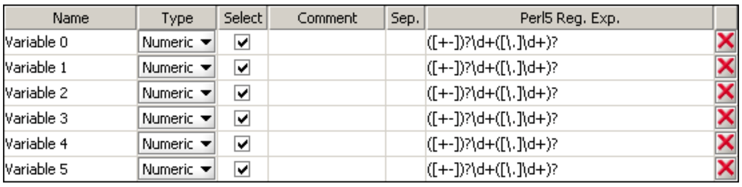
The application allows, by using regular expression, to utilize as input either log files of standard formats, as Apache and IIS, or log files modified by the user. The only important requirement on the input file structure is that every observation (i.e., the set of the variables that refer to one element/item considered) must end with '\n' character (new line).

To allow the application to be able to process correctly a log file, the user must describe:

- the observation structure, that is the number of variables that compose it.
- and for each variable the name, the type and the 'structure'.

These information are specified by the user through the input panel described in Section 4.1.2.

Regarding the number of variables it is enough to insert in the format table a number of rows equal to the number of elements that compose the observation (see Section 4.1.2). In Figure 4.51 the observation is composed of 6 variables.



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	
Variable 0	Numeric ▾	☑			([+-])?\d+([\.]\d+)?	☒
Variable 1	Numeric ▾	☑			([+-])?\d+([\.]\d+)?	☒
Variable 2	Numeric ▾	☑			([+-])?\d+([\.]\d+)?	☒
Variable 3	Numeric ▾	☑			([+-])?\d+([\.]\d+)?	☒
Variable 4	Numeric ▾	☑			([+-])?\d+([\.]\d+)?	☒
Variable 5	Numeric ▾	☑			([+-])?\d+([\.]\d+)?	☒

Figure 4.51: Format table after the insertion of the number of variables

The definition of the single variable requires, in addition to the name, a comment, its selection in order to be considered in the analysis or not, its type, the description of the possible delimiter (Separator) and of the regular expression that characterizes it.

The type has to be chosen among the *numeric, string o data.* The delimiter defines (if it exists) the character or the characters that delimit the variable's value. The regular expression (Perl 5) defines the format with which the variable value is recorded in the log file.

A concise description of the format used for the regular expressions (Perl 5) follows. The purpose of the definition of the regular expression is to specify the pattern that the variable values must have and therefore (together with selected type) to allow the application to process correctly its value. For a complete description of the regular expressions see a Mastering regular expressions book. In this section we will describe some options useful to define in a simple way the pattern of the variables values. A list of operators follows:

+ : it indicates that there are one or more instances of the preceding character (a+ ⇒ a,aa,aaa,...)

[ ]: it defines a pattern that allows to utilize as match element one of the values within square brackets (a[bcd]e ⇒ abe, ace, ade)

***** : it indicates that there are zero or more instances of the preceding character (ba* ⇒ b,ba,baa,baaa,...)

? : it indicates that there are zero or one instance of the preceding character (ba? ⇒ b,ba)

ˆ : it has two different meanings depending on the use. The former permits to require that a pattern starts with a particular character (ˆabc ⇒ abcdef,abc fg,...). If it is utilized within square brackets as first character, it allows to match any character that doesn't belong to the defined group ([ˆa] ⇒ bcd,wedfgr,...)

() : it defines a pattern that permits to use as match element the string that is contained between brackets ((ab)+ ⇒ ab,abab,ababab,...).

A list of *Escape Sequences* used to identify intervals of characters follows:

\d : it identifies whichever number [0-9].

\D : it identifies whichever thing except a number [ˆ0-9].

\w : it identifies whichever character or number [0-9a-zA-Z].

\W : it identifies whichever thing except a character or a number [ˆ0-9a-zA-Z].

\s : it identifies a *white space* character [\t\n\r\f].

\S : it identifies whichever thing except a *white space* character [ˆ\t\n\r\f].

. : it identifies whichever character except the *new line* character '\n'.

Combining in an opportune way the characters, the options and the *Escape Sequences* it is possible to define correctly the regular expression for each variable.

In the specific of regular expressions in Perl 5 there are some characters that are considered *metacharacters*. Some of them are the same operators defined before. In this case, to use such characters as elements in the expression is enough to precede them with the \ character.

The delimiter identifies one or more characters that delimit the value of a variable. In the log file of the Apache format the page URL requested by the user contains characters and spaces and is delimited by "". A simple definition of the *\w+* type regular expression could produce a wrong result because the value recovered by application would finish at the first met space and not at the closure ". To avoid this problem it is possible to utilize the delimiters so that by using the expression *.+* and specifying as delimiter " the result will be correct.

Concerning the Apache log file, and more precisely the *timestamp* field, it is possible to show another use of the delimiter. The value is contained between characters '[' e ']' (e.g, *[12/Dec/2001:00:00:03 +0100]*) and by using the field delimiter it is possible to specify as regular expression the following:

\d\d/\w\w\w/ \d\d\d\d:\d\d:\d\d:\d\d

This expression doesn't specify the last part of the field (*+0100*) but it is correct in any case because specifying the delimiters it is possible to define the regular expression in order to match only a part of the variable value.

4.2.1 Example of a format definition

Let's consider a file to be analyzed with the following structure of the observations:

String without spaces 1000 0.0876 -12.663 "String with spaces" [12/Dec/2001:00:00:03 +0100]

The variables are 6 and their definition (Figure 4.52) is the following:

String without spaces : we can define the regular expression in different ways, as for example *[a-zA-Z]+* if the string doesn't contain numbers or *\w+* to indicate whichever string of numbers and characters.

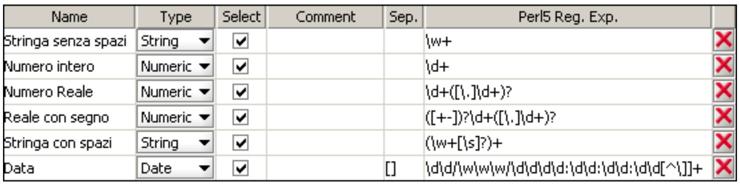
Integer number : we define the regular expression as a sequence of one or more numbers and therefore as *\d+*.

Positive decimal : in this case there is the possibility to have decimal digits so that we define the regular expression as a sequence of one or more digits (*\d+*) that may have the decimal separator and a further succession of one or more digits. The result is therefore *\d+([\.]\d+)?*.

Decimal with sign : in this case in comparison to the previous case there is the possibility that at the beginning of the number there is a sign +, - or no sign. Therefore it is enough to add at the beginning of the regular expression *([+-])?*.

String with spaces : if we utilize the regular expression previously defined *\w+* we would have a problem because the value recovered from the log file would be "String that is not correct. The best solution is to utilize the delimiter, in this case ", and to utilize therefore the definition *.+*.

Date : even in this case by using the delimiter (*[]*) we can define the following regular expression \d\d/\w\w\w/ \d\d\d\d:\d \d:\d\d:\d\d that may be used to recover the only information "12/Dec/2001:00:00:03" leaving out the remaining part.



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	
Stringa senza spazi	String	☑			\w+	✗
Numero intero	Numeric	☑			\d+	✗
Numero Reale	Numeric	☑			\d+([\.]\d+)?	✗
Reale con segno	Numeric	☑			([+-])?\d+([\.]\d+)?	✗
Stringa con spazi	String	☑			(\w+[\s]?)+	✗
Data	Date	☑		[]	\d\d/\w\w\w/\d\d\d\d:\d\d:\d\d:\d\d[^\]]+	✗

Figure 4.52: Example of the definition of a file format

4.2.2 Example of the definition of the Apache log

Now we will describe the format of the Apache log. Each observation has the following structure:

```
151.29.12.105 - - [12/Dec/2001:00:00:10 +0100] “GET /mappe HTTP/1.1” 200 313 www.polimi.it “http://www.polimi.it/”
“Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)” “-”
```

A brief explanation of the regular expressions used (Figure 4.53) follows:

1. for the Ip address of the request we define a sequence of one or more digits separated by a point and therefore the regular expression is \d+\.\d+\.\d+\. \d+.

2. in most cases no value is set, and therefore it is possible to define exactly the string that we want to recover from the file through - - (it is advisable not to select this variable at the moment of loading).

3. for the date we use the same definition described in the previous example or more simply by selecting the date type in the concerning box, the application supplies a default definition that is correct too.

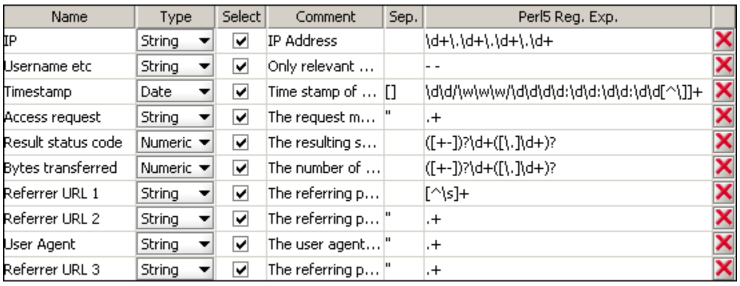
4. it is a string, delimited by ”, that can contains within some spaces characters and numbers. The best definition uses the delimiter and the regular expression .+.

5. it is an integer number and as we have seen before we can utilize a more simple regular expression \d+ o ([+-])?\d+([\.] \d+)?.

6. even in this case it is an integer number and as we have seen before we can utilize the regular expression \d+ o ([+-])?\d+([\.] \d+)?.

7. in this case the string is not delimited by any particular character and it doesn’t contain any spaces so that we can utilize a simple regular expression \w+ or ones little bit different as [ˆ\s]+.

8. these last three strings are all delimited by the character ” and they can contain some spaces, characters or numbers and therefore we must utilize the regular expression .+.



Name	Type	Select	Comment	Sep.	Perl5 Reg. Exp.	
IP	String	☑	IP Address		\d+\.\d+\.\d+\.\d+	☒
Username etc	String	☑	Only relevant ...		- -	☒
Timestamp	Date	☑	Time stamp of ...	[]	\d\d/\w\w\w/\d\d\d\d:\d\d:\d\d:\d\d[^\]]+	☒
Access request	String	☑	The request m...	"	.+	☒
Result status code	Numeric	☑	The resulting s...		([+-])?\d+([\.]\d+)?	☒
Bytes transferred	Numeric	☑	The number of ...		([+-])?\d+([\.]\d+)?	☒
Referrer URL 1	String	☑	The referring p...		[^\s]+	☒
Referrer URL 2	String	☑	The referring p...	"	.+	☒
User Agent	String	☑	The user agent...	"	.+	☒
Referrer URL 3	String	☑	The referring p...	"	.+	☒

Figure 4.53: Apache log file format definition

Appendix A

Basic Definitions

Customer : or *job* or *transaction* is the element that will require service from our network. For example, it can be an *http request*, a *database query*, or an *ftp download command.*

Class : a group of elements that are statistically equal. The customers of a class have the same workload intensity and the same average service demands.

Station : or service center, it represents a system resource. Customers arrive at the station and then, if necessary, wait in queue, receive service from server then depart.

Bottleneck Station : a station with the highest utilization

Potential Bottleneck Station : a station that can become bottleneck for some feasible network population

Dominated Stations : a station P is dominated if exist al least a station Q whose demand for each classes of custumer are highest or equal (but at least one highest) than those of P.

Masked-off Stations : a station that is not a Potential Bottleneck or a Dominated station.A Masked-off station may exhibit the largest queue-lungth and hence the highest response time.

Workload Intensity (λ or N) : rate at which customers of a given class arrive at the system. Each class has its own workload intensity. If a class is closed the constant *number of customers* N that are present in the system must be provided, if a class is open the *arrival rate* λ of customers to the system is requested.

Population Mix (β_i) : the ratio between *number of customers* of closed class i and the total number of customers in the system: $\beta_i = N_i / \sum_k N_k$

Saturarion Sector : a interval of Population Mix in witch one, two ore more stations are both bottleneck.

Service Time (S_k): average time of service required at each visit to resource k; it is computed as the ratio of the busy time to the number of system completions. It is an alternatively way to describe the service requirement.

Visit (number of -) (V_k) : average number of visits that a customer makes at station k during a complete execution; it is computed as the ratio of the number of completions at resource k to the number of system completions. If resource k is a delay center representing a client station, it is a convention assign a unitary value to number of visits to this station.

Service Demand (D_k) : average service requirement of a customer, that is the total amount of service required by a complete execution at resource k. In the model it is necessary to provide separate service demand for each pair service center-class. It is given by $D_k = V_k * S_k$.

Throughput (X_k) : rate at which customers are executed by station k, that is the number of completions in a time unit.

Queue length (Q_k) : average number of customers at station k, either waiting in queue and receiving service.

Response Time (R_k): average time interval between the instant a customer arrives at station k and the instants it terminate its servicing.

Residence Time (W_k) : average time that a customer spent at station k during a complete interaction with the system. It includes time spent queueing and time spent receiving service. It does not correspond to *Response Time* R_k of a station since $W_k = R_k * V_k$.

Utilization (U_k) : proportion of time in which the station k is busy or, in the case of a delay center, is the average number of customers in the station (see Little Law [Lit61]).

System Throughput (X) : rate at which customers perform an entire interaction with the system. This is the aggregate measure of *Throughput*: $X = X_k/V_k$.

System Response Time (R): correspond to the intuitive notion of response time perceived by users, that is, the time interval between the instant of the submission of a request to a system and the instant the corresponding reply arrives completely at the user. It is the aggregate measure of *Residence Times*: $R = \sum_k W_k$.

Average number of customers in the system (N): the average number of customer in the system, either waiting in queue or receiving service. It is the aggregate measure of *Queue Length*: $N = \sum_k Q_k$.

Bibliography

[BBC+81] G. Balbo, S.C. Bruell, L. Cerchio, D. Chialberto, and L. Molinatti. *Mean value analysis of closed load dependent queueing networks.* Dipartimento di Informatica, Universit di Torino, Oct. 1981.

[BCMP75] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. *Open, Closed and Mixed Networks of Queues with Different Classes of Customers.* J. ACM, April 1975.

[GC04] Giuseppe Serazzi Giuliano Casale. *Bottlenecks Identification in Multiclass Queueing Networks using Convex Polytopes.* 2004.

[Lit61] John D. C. Little. *A Proof of the Queueing Formula $L = \lambda W$. Operations Research*, 9:383–387, 1961.

[LZGS84] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K. Sevcik. *Quantitative System Performance.* Prentice-Hall, 1984.

[RL80] M. Reiser and S.S. Lavenberg. *Mean-Value Analysis of Closed Multichain Queuing Networks.* J. ACM, Vol 27, No 2, pp 313-322, April 1980.