DICE-FG input data format

To standardise the use of the estimation algorithms, we have adopted a common data format, from which each algorithm can select the data it requires to perform the estimation. We assume the data has been or is being collected for an application that provides a number of services, grouped in service classes.

There are a total of *K* different service classes and *M* different resources. Data is collected either by averaging in time windows, or for individual requests, or both, and can be specific to a class or aggregate. Data is associated to a subgraph of *E*<=*M* resources. If *E*=1, the data is stored in *ResourceDataFile*, conversely if it is associated to a subgraph of multiple resources (*E*>1) the data is stored in *GraphDataFile*. We discuss the two cases separately.

Resource data

The data format is a data structure (MATLAB cell array) with 11 rows and *M(K+1)* columns, representing M groups of (*K + 1*) columns. The *i*-th group of (*K+1*) columns represents the measurements for the *i*-th resource and uses the first *K* columns to describe data for each service class, while the last column is reserved for aggregate data. The column index of class *r* at resource *i* is therefore **idx=(*i*-1)*(K+1)+*r*,** while for the aggregate data at resource *i* it is **idx=*i*(K+1)** .

For each column, the information provided in each row is the following:

Row ID	***Metric type***	***Unit of measure***	***MetricName***	***Description***
1	**Sampling timestamp**	milliseconds	**ts**	Holds the timestamps corresponding to the end of each sampling interval
2	**Utilization**	N/A, in [0,1]	**util**	Holds the average CPU utilization for each sampling window. Typically only overall CPU utilization is collected, thus only the column *K +1* will hold an array, while the other columns will be empty.
3	**Arrival timestamps**	milliseconds	**arvT**	Holds the timestamps of the arrival of each request to the resource.
4	**Response time**	seconds	**respT**	holds the observed response times (departure time minus arrival time) of each request
5	**Average response time**	seconds	**respTAvg**	Holds the mean response time of the requests processed in each sampling window. If no requests of a given class are processed in a sampling interval, the corresponding entry in the array is set to zero.
6	**Average throughput**	jobs/second	**tputAvg**	Holds the throughput observed for each service class in each sampling window. The throughput is computed as the total number of requests processed in the sampling interval, divided by the length of the interval (in seconds).
7	**Departure times**	milliseconds	**depT**	Holds the timestamps of the departure of each request from the resource.

8	**Queue-length**	jobs	**qlen**	Holds the actual queue-length (number of jobs in the system) for each request arriving in each sampling period.
9	**Average queue length**	jobs	**qlenAvg**	Holds the average queue-length (number of jobs in the system) for each service class in each sampling period.
10	**Memory usage**	kilobytes	**mem**	Holds the memory usage in each sampling window. More accurate results can be obtained if the memory is computed as the total memory usage minus the memory allocation due to operating system or other services running in the background.
11	**Average memory usage**	kilobytes	**memAvg**	Holds the average memory usage in each sampling window. More accurate results can be obtained if the memory is computed as the total average memory usage minus the memory allocation due to operating system or other services running in the background.

Graph data

At the moment, DICE-FG supports only graph-wise estimation for metrics recorded on the entire set of resources, i.e., *E=M*. For example, the end-to-end response time, the system throughput, and the total number of jobs in the system. The specification of data is again based on columnar data, with the first 11 rows as for the resource data. However, the successive rows are as follows:

12. **Routing matrix**: a probability matrix of order *(M+1)* specifying the route of requests across the network of resources, with the upper-left submatrix of order *M* with entries representing the routing across *M* resources, and the remaining entries representing flows from/to the outside world (open topology) or the think time of the users (closed topology). At present, no class switching is allowed through a routing path.

13. **Consumption matrix**: a binary matrix with *M* rows and *K* columns. If element *(m,k)* is set to 1, then it is assumed that a visit of a job to resource *m* consumes resources as specified by the resource data for class *k* at resource *m*. It is possible to set to 1 several classes *k1, k2,..,kr* on the same resource *m*, in this case all the class data of the *r* classes will be summed to determine the resource consumption at resource *m*, after appropriate weighting for the routing probabilities to this resource.

Given the above routing matrix, the information provided in the first 11 rows is interpreted as follows:

MetricName	***Interpretation in GraphData***
ts	Timestamps at the end of the sampling periods. For aggregate measurements, the sampling period ends when all the required metrics have been collected.
util	Percentage of admitted jobs in the sampling period, assuming a limited number of jobs can be admitted in the subgraph.
arvT	Timestamps of arrivals from node *M+1* into any of the M resources.
respT	Response time between arrival from node *M+1* to return to node *M+1*, for each request visiting the *M* resources (or a subset thereof).
respTAvg	Averaged value of response times over sampled windows.

tputAvg	Mean departure rate from any of the M resources to node M+1.
depT	Timestamps of arrivals from any of the M resources to node M+1.
qlen	Number of jobs observed in the system at the end of the sampling period.
qlenAvg	Average number of jobs observed in the system.
mem	Memory usage summed across the M resources as seen at arrival instants of jobs from node M+1.
memAvg	Cumulative average memory usage summed across the M resources as seen at the end of the sampling period.

Specifying DICE-FG input data via JSON

Two functions are provided with DICE-FG to convert the common data format to/from JSON. The common data format is composed by the **data, resources** and **classes** cell arrays. These can be converted to JSON from MATLAB using the command:

fg2json('hmr', resdata, graphdata, resources, resclasses, graphclasses)

Where ‘hmr’ is a use specified text prefixed. This will create three JSON files: *hmr-data.json*, *hmr-resources.json* and *hmr-classes.json*. To reload these files into the MATLAB environment, DICE-FG uses the command:

[resdata, graphdata, resources, resclasses, graphclasses] = json2fg(folder, 'hmr')

Where folder is the path to the folder containing both JSON files, e.g., the output of MATLAB *pwd* command for the current directory. JSON files can also be specified directly, without use of MATLAB. Examples are provided with this tool release.