Using AMIR to Generate Schedules

This scripts shows how to use the provided MATLAB functions to generate schedules for the techniques discussed in the paper. Please note that to generate schedules based on AMIR in this MATLAB implementaion, CVX and MOSEK are required. Please refer to http://cvxr.com/cvx/licensing/ on how to install them.

Set the environment to allow 2000 times of recursion. This may be needed for traversing large M in the recursive function walkMarkovChainLookAheadRandom.

```
set(0,'RecursionLimit',2000);
```

Service Demand Distributions

Loading sample distributions for 3 resources and 3 session classes. Other distributions may be used instead.

```
load('logNormalDistProbs.mat');
```

Arrival vector A

Sample A with three session classes:

```
A = [80 15 30];
```

P^0 Vector Calculations

Calculating P^0 vector as well as the mean service demand at each resource:

```
[PR1, meanServiceR1] = calculateProbability(A, probDistsR1)
```

```
PR1 =

    0.4206    0.0083    0.0391
meanServiceR1 = 23.4000
```

```
[PR2, meanServiceR2] = calculateProbability(A, probDistsR2)
```

```
PR2 =

    0.0095    0.9964    0.4151
meanServiceR2 = 18.7200
```

```
[PR3, meanServiceR3] = calculateProbability(A, probDistsR3)
```

```
PR3 =

    0.0064    0.3511    0.9647
```

```
meanServiceR3 = 10.2400
```

Considering the mean service demands, R1 is the bottleneck resource.

Burstiness B^O Minimization for the Bottleneck Resource

Selecting an order for burstiness minimization:

```
O = 2;
```

Finding the optimal schedual M and its burstiness B for order O

```
[M, B] = optimalScheduleRecursive(A, A, PR1', PR1, 1, O, 1);
```

Generating Schedules from M^O

Number of random schedules to generate from M^O:

```
numberOfSchedules = 1;
```

Traversing the M^O to generate the schedules:

```
schedules = generateMultipleOptimalSchedules(M, numberOfSchedules)
```

```
schedules =

     2     1     1     2     1     1     3     1     1     3     1     3     1
```

Removing additional O session(s) from the circular schedule (refer to fig. 4)

```
schedulesAdjusted = schedules(:, 1:end-O)
```

```
schedulesAdjusted =

     2     1     1     2     1     1     3     1     1     3     1     3     1
```

The schedule shows the class of sessions and their order. For example, the schedule (3 1 1 2 ...) shows a session from class 3 is followed by a session from class 1 that is also followed by a session from class 1 and so on.

Generating SJF, LJF, Random, and Greedy Schedules

SJF:

```
SJFSchedule = generateShortestOrLongestFirstSchedule( A, probDistsR1, 'shortestFirst' )
```

```
SJFSchedule =

     2     2     2     2     2     2     2     2     2     2     2     2     2
```

```
% LJF:
LJFSchedule = generateShortestOrLongestFirstSchedule( A, probDistsR1, 'longestFirst' )
```

```
LJFSchedule =

     1     1     1     1     1     1     1     1     1     1     1     1     1
```

```
% Random:
randomSchedules = generateMultipleRandomSchedules( A, numberOfSchedules)
```

```
randomSchedules =

     1     3     1     1     2     1     1     3     1     1     1     1     1
```

```
% Greedy
greedySchedule = generateGreedyScheduleNew( A, PR1)
```

```
greedySchedule =

     1     2     1     2     1     2     1     2     1     2     1     2     1
```