In this homework, you will run simulations to evaluate the performance of task-assignment algorithms. In particular, we are interested in finding out how successful the basic bin-packing algorithms are in finding suitable task allocations.

(1) First, find yourself a good random number generator. This is a piece of code which generates numbers that are uniformly distributed over the interval [0, 1]. Don't use the one that comes with your system\(^1\). Do some research to pick a good generator (there is a large literature that is available on the web; just use your favorite search engine to find it). Read up on the criteria that are used to evaluate such generators, and list them, along with the generator you picked.

(2) We now want to generate task sets that can be feasibly allocated and scheduled on a set of \(n\) processors, where \(n\) is an input. We'll start by constructing \(n\) task sets, each with total processor utilization no greater than 1. This guarantees that at least one assignment exists, under which EDF can meet all deadlines. Then, we'll take the union of all these task sets and present this union to our task-assignment algorithm. Write software to generate sets of \(m\) tasks each, with total utilization \(u\). Both \(m\) and \(u\) are inputs to the program. The utilization of each task should be chosen randomly with the proviso that the sum of the utilizations in any given set is equal to \(u\).

(3) Now, generate the union of all these task sets. You'll have a total of \(nm\) tasks.

(4) Sort the tasks in ascending order of their utilizations. Present this list of tasks to a first-fit allocation algorithm and check if it is able to successfully allocate them.

(5) Sort the tasks in descending order of their utilizations, and present this list to a first-fit allocation algorithm.

(6) Repeat (4) and (5) above for a best-fit allocation algorithm.

(7) For each set of parameters (specified below), compute the fraction, \(f\), of instances that can be scheduled successfully. Also generate the 95% confidence intervals: the half-width, \(h\), of these intervals is equal to \(1.96\sqrt{f(1-f)/r}\), where \(r\) is the number of runs for that set of parameters. The 95% confidence interval is \([f-h, f+h]\). You should make enough runs to ensure that \(h \leq 0.05f\).

(8) Plot your results, with the average utilization per processor varying from 0.5 to 1.0

\(^1\)This is a general principle that you should follow in any simulations you write: never use a random-number generator unless you've satisfied yourself that it is a good one. Just because it "comes with the system" is no guarantee of that.
on the x-axis and the success rate, f, on the y axis. (If the curve is flat for most of that interval, you may want to provide additional plots zooming in on the "interesting" regions). Assume that you have a total of $n = 4$ processors and plot curves for $m = 3, 5, 7, 9$ (i.e., the total number of tasks will be 12, 20, 28, 36).

(9) Investigate the impact of varying the number of processors, $n$. Plot results, with $n$ on the x-axis and $f$ on the y-axis; varying $n$ from 2 to 10, in steps of 1. Plot curves for the following average utilizations per processor: 0.9, 0.95, 0.975. Assume 3$n$ tasks in all.

**Organization of Homework:** I suggest typing your homework, using any text-processing program you have access to. On-campus students have ready access to LaTeX, which is probably the best widely-available software for technical writing. (An obvious alternative is Word). The default can be simply to write it up as an ASCII file that you can then print out for submission.

Your homework should be organized as follows.

- Section 1 should contain information on the random number generator that you picked, with a description (no longer than one page) of the various ways of testing such generators. (You don't have to have run these tests on the generator you pick, so long as you can cite some reference that vouches for the quality of the generator).

- Section 2 should present your plots for the first-fit algorithm. On-campus students should use some plotting package to generate them: gnuplot on Linux is probably the best; others are available as well. Off-campus students who don't have access to any such packages can simply draw these plots on graph paper, although doing so is tedious and should be avoided if at all possible.

Also, identify any trends in your plots and discuss them. This discussion should not be longer than about a page (in fact, about half a page should do).

- Section 3 should contain the plots and a discussion of the results for the best-fit algorithm. Compare the results of the first-fit and best-fit algorithms: provide plots comparing the first-fit and best-fit data for $m = 3, 9$ and $n = 4$ processors.

- In Section 4, discuss what changes you would make to your software to allow it to be used in generating a fault-tolerant assignment and schedule (recall our discussion of primary and ghost copies), to sustain the failure of up to $n_{sust}$ processors.
• The Appendix should include a listing of your source-code: this should be fully documented so that I can understand it.

Off-campus students: Homework should be mailed to C. M. Krishna, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003, or faxed to (413) 545-1227. Always keep a copy of everything you mail out.