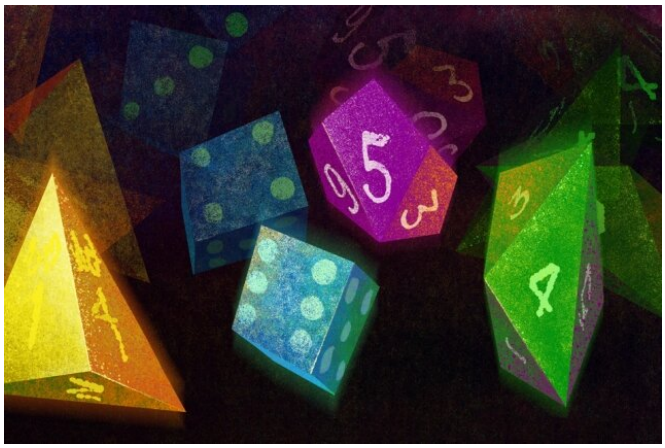
Algorithm quickly simulates a roll of loaded dice

29 May 2020, by Steve Nadis



A new algorithm, called the Fast Loaded Dice Roller (FLDR), simulates the roll of dice to produce random integers. The dice, in this case, could have any number of sides, and they are “loaded,” or weighted, to make some sides more likely to come up than others. Credit: Jose-Luis Olivares, MIT

The fast and efficient generation of random numbers has long been an important challenge. For centuries, games of chance have relied on the roll of a die, the flip of a coin, or the shuffling of cards to bring some randomness into the proceedings. In the second half of the 20th century, computers started taking over that role, for applications in cryptography, statistics, and artificial intelligence, as well as for various simulations—climatic, epidemiological, financial, and so forth.

MIT researchers have now developed a computer algorithm that might, at least for some tasks, churn out random numbers with the best combination of speed, accuracy, and low memory requirements available today. The algorithm, called the Fast Loaded Dice Roller (FLDR), was created by MIT graduate student Feras Saad, Research Scientist Cameron Freer, Professor Martin Rinard, and Principal Research Scientist Vikash Mansinghka, and it will be presented next week at the 23rd International Conference on Artificial Intelligence and Statistics.

Simply put, FLDR is a computer program that simulates the roll of dice to produce random integers. The dice can have any number of sides, and they are "loaded," or weighted, to make some sides more likely to come up than others. A loaded die can still yield random numbers—as one cannot predict in advance which side will turn up—but the randomness is constrained to meet a preset probability distribution. One might, for instance, use loaded dice to simulate the outcome of a baseball game; while the superior team is more likely to win, on a given day either team could end up on top.

With FLDR, the dice are "perfectly" loaded, which means they exactly achieve the specified probabilities. With a four-sided die, for example, one could arrange things so that the numbers 1,2,3, and 4 turn up exactly 23 percent, 34 percent, 17 percent, and 26 percent of the time, respectively.

To simulate the roll of loaded dice that have a large number of sides, the MIT team first had to draw on a simpler source of randomness—that being a computerized (binary) version of a coin toss, yielding either a 0 or a 1, each with 50 percent probability. The efficiency of their method, a key design criterion, depends on the number of times they have to tap into this random source—the number of "coin tosses," in other words—to simulate each dice roll.

In a landmark 1976 paper, the computer scientists Donald Knuth and Andrew Yao devised an algorithm that could simulate the roll of loaded dice with the maximum efficiency theoretically attainable. "While their algorithm was optimally efficient with respect to time," Saad explains, meaning that literally nothing could be faster, "it is inefficient in terms of the space, or computer

memory, needed to store that information." In fact, the amount of memory required grows exponentially, depending on the number of sides on the dice and other factors. That renders the Knuth-Yao method impractical, he says, except for special cases, despite its theoretical importance.

FLDR was designed for greater utility. "We are almost as time efficient," Saad says, "but orders of magnitude better in terms of memory efficiency." FLDR can use up to 10,000 times less memory storage space than the Knuth-Yao approach, while taking no more than 1.5 times longer per operation.

For now, FLDR's main competitor is the Alias method, which has been the field's dominant technology for decades. When analyzed theoretically, according to Freer, FLDR has one clear-cut advantage over Alias: It makes more efficient use of the random source—the "coin tosses," to continue with that metaphor—than Alias. In certain cases, moreover, FLDR is also faster than Alias in generating rolls of loaded dice.

FLDR, of course, is still brand new and has not yet seen widespread use. But its developers are already thinking of ways to improve its effectiveness through both software and hardware engineering. They also have specific applications in mind, apart from the general, ever-present need for random numbers. Where FLDR can help most, Mansinghka suggests, is by making so-called Monte Carlo simulations and Monte Carlo inference techniques more efficient. Just as FLDR uses coin flips to simulate the more complicated roll of weighted, many-sided dice, Monte Carlo simulations use a dice roll to generate more complex patterns of random numbers.

The United Nations, for instance, runs simulations of seismic activity that show when and where earthquakes, tremors, or nuclear tests are happening on the globe. The United Nations also carries out Monte Carlo inference: running random simulations that generate possible explanations for actual seismic data. This works by conducting a second series of Monte Carlo simulations, which randomly test out alternative parameters for an underlying seismic simulation to find the parameter values most likely to reproduce the observed data. These parameters contain information about when and where earthquakes and nuclear tests might actually have occurred.

"Monte Carlo inference can require hundreds of thousands of times more random numbers than Monte Carlo simulations," Mansinghka says. "That's one big bottleneck where FLDR could really help. Monte Carlo simulation and inference algorithms are also central to probabilistic programming, an emerging area of AI with broad applications."

Despite its seemingly bright future, FLDR almost did not come to light. Hints of it first emerged from a previous paper the same four MIT researchers published at a symposium in January, which introduced a separate algorithm. In that work, the authors showed that if a predetermined amount of memory were allocated for a computer program to simulate the roll of loaded dice, their algorithm could determine the minimum amount of "error" possible—that is, how close one comes toward meeting the designated probabilities for each side of the dice.

If one doesn't limit the memory in advance, the error can be reduced to zero, but Saad noticed a variant with zero error that used substantially less memory and was nearly as fast. At first he thought the result might be too trivial to bother with. But he mentioned it to Freer who assured Saad that this avenue was worth pursuing. FLDR, which is error-free in this same respect, arose from those humble origins and now has a chance of becoming a leading technology in the realm of random number generation. That's no trivial matter given that we live in a world that's governed, to a large extent, by random processes—a principle that applies to the distribution of galaxies in the universe, as well as to the outcome of a spirited game of craps.

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology