Needle in a Haystack: Sampling Factorial Solution Spaces

Ben Draut
u1066504

February 9, 2019

Abstract

SweetPea, a language for experimental design, utilizes SAT-samplers to generate approximately uniformly sampled trial sequences. In practice, this approach does not scale to support realistic experimental designs in which the solution space grows factorially. We propose that alternative approaches be investigated to overcome this problem. This would include different encoding schemes as well as techniques for direct construction of a trial sequence based on its numerical index in the sequence of all solutions. A successful alternate approach will greatly improve the utility of SweetPea for real-world use cases.

Background

SweetPea is a domain-specific language, for designing experiments, that improves on the existing tooling. Prior to SweetPea, scientists cobbled together scripts that procedurally constructed trial sequences conforming to the intended design. In addition to being error-prone, this method produced biased results. While SweetPea is still useful, it has not yet realized the aspiration of guaranteeing approximately uniformly sampled trial sequences (unbiased) for realistic designs. SweetPea encodes the experimental design as a SAT formula, and then relies on a SAT-sampler, such as `unigen`, to sample solutions to the formula. While this works reasonably well for small designs, it does not scale to realistic design sizes.

The primary hurdle is that the SAT formula encodes all possible permutations of every valid trial sequence as a distinct solution. As a result, the size of the solution space grows factorially in the sequence length. For example, a small but realistic design yields a sequence with 36 trials, or roughly $36! \approx 2^{138}$ distinct solutions. While SAT-*solvers* are still able to quickly find individual solutions to these formulae, SAT-*samplers* have proven unable to cope with solution spaces of this magnitude, despite the overall size of the formula being small.

One of two alternate approaches, or a hybrid, may allow SweetPea to successfully sample sequences from realistic designs:

- Alternate encoding scheme. Although we cannot reduce the size of the solution space, perhaps we can reduce the burden placed upon the SAT-sampler. Can we construct an encoding that would identify categories, or segments of the solution space, followed by further sampling in code?
- Direct construction approach. An experimental design describes in detail what constitutes a valid solution. Using this information, can we compute the exact number of solutions, n, and develop a polynomial-time algorithm that can construct the i^th solution? This would shift the burden of guaranteeing uniformity to simply uniformly sampling the natural numbers from 1 to n for the desired number of trial sequences.

Proposal and Outline

I propose a thesis that meets the following objectives:

1. Report on the observed performance of SAT-samplers we've used (`unigen` and `KUS`), including metrics on the formula size and solution space for multiple experiment designs.

2. Propose at least one alternate encoding scheme that doesn't require a SAT-sampler to sample the full solution space.
3. Develop an algorithm to compute the precise number of solutions for a given design.

Chapter Outline:

1. Motivation - This section will contain a brief overview of SweetPea, as well as a detailed explanation of the factorial explosion problem in the current encoding. For details of the current encoding, it will refer back to the original SweetPea paper.
2. Sampler Benchmarks - This section will present 3-4 experiment designs of increasing complexity, as well as key metrics of the associated encoding, including the size of the formula, and the estimated number of solutions. It will describe how both samplers, `unigen` and `KUS` behave when running against each design. It will also include benchmarks for the latest version of `ApproxMC`, which is being used in the next version of `unigen` and is a good indicator of how well it will perform for our use cases. This section should convince the reader of the need for another strategy.
3. Alternate Encoding Schemes - This section will introduce one or more possible encoding schemes that would reduce the burden on the SAT-sampler. It will describe alternative sampling models that may relax the uniformity constraints to improve performance, as well as the extent of the relaxation.
4. Solution Counting - This section will describe an algorithm that computes the precise solution count for a particular design. It will prove correctness based on based on existing techniques for counting sets. It will highlight insights that may lead to a construction function for directly producing individual trial sequences.
5. Related Work - This section will reference other current works related to SAT encodings, sampling, and solution counting.
6. Future Work - This section will enumerate the next steps to be undertaken, including implementation of an alternate encoding scheme, milestones to progress towards a construction function for an arbitrary sequence number, and benchmarks demonstrating the improvements over prior attempts.
7. Conclusion

Conclusion

A complete thesis following the form and objectives described here would lay a foundation for further work to overcome the problem of sampling such an enormous solution space.