1 Introduction

Coupling from the past is a Markov Chain algorithm which is helpful for generating, in principle, a perfect sampling from a stationary distribution [1]. Since we are interested in observing from a sample distribution over the partitions that can fit inside of a $n \times n \times n$ box, we use what those authors refer to as Coupling From the Past with stochastic flows in discrete time. For computational reasons we keep n small. In the code, we often use $n = 3$ as the case for demonstrating the proof-of-concept. This allows for convergence within 1000 or so transitions. In order to determine the worst-case convergence between any two partitions living inside the cube of side length n, we consider the time it takes for the empty partition and the full partition to converge. From that point on, a transition will be valid for one partition if and only if it is valid for both partitions, as they consist of the same blocks. This means that after the first time-step where the two partitions are equal, they will always be equal for every subsequent iteration since they will be composed of the same cubes, and as such a move representing an addition is valid for one if and only if it is a valid addition for the other.

The algorithm for implementing Coupling From the Past is quite simple. We present the pseudo-code below (see Algorithm 1). Assume the preconditions of the algorithm are that p_1 and p_2 are two (distinct) partitions, and M is some sufficiently large real number. Furthermore assume that GenerateMove is a function that generates a single transition, and DoMove is a function that does the move on the partitions

Informally, this algorithm represents building a set of moves such that if we let those moves act on our partition, then eventually the partitions will both be transformed into an identical partition.

The paper will hereafter be split into two major sections. In the first, 2 we discuss the code

Algorithm 1 CFTP(p_1, p_2, M)

```
MoveList ← $\emptyset$
temp$_1$ ← $p_1$                      ▷ Store partitions in memory
temp$_2$ ← $p_2$
while temp$_1 \neq$ temp$_2$ do
  temp$_1$ ← $p_1$                    ▷ Reset the temporary partitions for the next iteration
  temp$_2$ ← $p_2$
  for count = 0, count < $M$, count + + do
    MoveList ← {GenerateMove()}+ MoveList ▷Prepend $M$ moves to the list of moves
  end for
  for move in MoveList do
    (temp$_1$, temp$_2$) ← DoMove(temp$_1$, temp$_2$, move)  ▷ Execute the list of moves on the given
                                                     ▷ partitions. After this line, we test if the
                                                     ▷ set of moves is sufficient for convergence
  end for
end while
```

convince the reader that we have faithfully implemented the Coupling From the Past algorithm, and in the second section, 3, we present rigorous arguments to convince the reader that this sampling process achieves the desired results. Namely that this model accurately samples the space of transitions with a uniform distribution (therefore representing a bounded random walk), and therefore if we grant that random number generation in Mathematica is sufficiently random, then we will have demonstrated a method for bounding in expectation the number of moves required for the convergence of any two partitions within a particular cube.

2 Implementing Coupling From the Past in Code

We present the argument that transitions to valid moves are computed correctly in order to convince the reader that the code produces a meaningful output that accurately describes Coupling From the Past.

```
ValidMove[location_List, move_, partition_List] := Module[{temp},
  (* If add, try to add the block,
  and check if it is still a valid partition. If remove,
  try to remove the block and it if it is still a valid partition \
then it was a legal move. *)
  temp = partition;
  row = location[[1]];
  column = location[[2]];
  height = location[[3]];
  addFlag = move;
  currentHeight = temp[[row, column]];
  If[addFlag == 1, If[height == currentHeight + 1,
    (* In this case, it was a valid move to add a block, so do so:*)
      temp[[row, column]] += 1;
    Result = IsValidPartition[temp];
    (* Undo any potential problems wrt maybe pass-by-reference? *)
     temp[[row, column]] -= 1;
    Return[Result];, (* Here it was not a valid move to add a block*)
    Return[False]],
   If[addFlag == -1,
    (* In this case, we are to remove a block.
    We need to test if the height is the same as current height*)
    If[height == currentHeight && height > 0,
     temp[[row, column ]] -= 1;
     Result = IsValidPartition[temp];

     temp[[row, column]] += 1;

     Return[Result]]

    , Return[False]]

   ];

  Return[False]

  ]
```

The author would like to acknowledge [2] for their Mathematica code implementing the graphical representation of plane partitions, as well as the inspiration for the matrix representation of the partitions.

We presuppose that a move specified a location where the action is happening, which is essentially a 3-coordinate tuple representing the row, column, and height of the action, and whether or not the action is to add a block or to remove a block. Then in the case of adding a block, we test if the block is already included in the partition, and if it is on the top layer of the partition (there are no blocks in the partition above that point), then we try to add the block to the partition and test if that yields a valid partition. If so, then we have specified a valid move. We store the partition in a temporary variable and make the change, and then reset the partition as it was before. This is intended to combat any problems that may arise if Mathematica is storing the variable as a pointer instead of an array, which would cause the actual partition to be overwritten. We are proceeding with an abundance of caution.

Now the same argument is applied to the case of removing a block. If the location specifies a block that is on top of our partition, then we remove the block, and check if the resulting partition is still a valid partition. Since we never execute a move without first checking if it generates a valid partition, our transformations will be an operator on the space of valid plane partitions living inside of an $n \times n \times n$ cube.

It turns out that this code runs with reasonable efficiency despite initial concerns surrounding its memory usage. It is able to execute approximately 2000 transitions (as is required for the $n = 3$ case) quickly enough as to appear instantaneous.

3 Mathematical Arguments

In this section, we essentially provide an expository analysis of the rich literature surrounding the theoretical background on the topic. For a deeper understanding of the results, we refer the reader to the original paper introducing the concept of Coupling From the Past, [3], which is the main source for this entire section.

At first glance, perhaps the most unexpected aspect of CFTP is the necessity of prepending moves to the current set of moves rather than appending, which is computationally thought of as looking backwards in time rather than forwards (thus coupling *from the past*). Historically, Propp and Wilson argue that these transformations are "substantially" more efficient with respect to runtime. It turns out that this is not necessarily the case with plane partitions, as generating the move sequence is constant time in either direction, and as such it requires O(numMoves) for both time and space in order to generate sufficient number of moves as to converge in either direction. We can't save any time with clever tricks in either direction since we simply must generate and perform the operations. The more mathematical argument as to why we require coupling to be from the past is significantly more clever.

Note that with probability 1, the process will terminate. We observe termination since at each state, each transition occurs with uniform probability so the algorithm failing to converge would be restricted to a set of density 0 in the set of infinite paths through the plane partitions (this ends up being a rather interesting word-avoidance problem). In fact, we can actually modify the underlying rigorous argument here to show that taking the set of previous moves to be infinite, we observe the stationary distribution. The stationary distribution is essential to our process, as now we have an infinite Markov process, and so we can look at a sufficient tail of the process, and that will not impact the final outcome, since we know that it converges. Here convergence will refer to the rather weak notion that the two partitions become indistinguishable after some finite set of operations. We can take this slightly further, noting that since we are drawing from the tail of a stationary distribution, now as long as the transformation is ergodic, then the starting state nor the amount of time used in the simulation will influence the final state. Therefore we are able to avoid many of the most challenging theoretical hurdles presented by standard Monte Carlo simulations.

Now we assumed above that the transformation is ergodic, which is one of the requirements of coupling from the past, as presented in its original paper. Now we must verify that we actually observe a stationary distribution. Observe that there are the same number of transitions into and out of each state. As a result, when the transitions are selected randomly, there cannot be any bias towards or away from any particular state, the distribution of states will tend to be uniform. In essence, what we have just described is a stationary state. Assuming some very mild conditions, it turns out that if a stationary state exists, then it must be unique, and in particular since we are modeling the tail of an infinite Markov process, we ensure that we are observing the convergence of the model to this particular stationary state.

References

[1] J. Propp and D. Wilson, "Coupling from the past: A user's guide.," *Microsurveys in discrete probability*, vol. 41, pp. 181–192, 1997.

[2] E. W. Weisstein, *Plane partition.* [Online]. Available: `https://mathworld.wolfram.com/PlanePartition.html`.

[3] J. G. Propp and D. B. Wilson, "Exact sampling with coupled markov chains and applications to statistical mechanics," *Random Structures & Algorithms*, vol. 9, no. 1-2, pp. 223–252, 1996. DOI: `https://doi.org/10.1002/(SICI)1098-2418(199608/09)9:1/2<223::AID-RSA14>3.0.CO;2-O`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291098-2418%28199608/09%299%3A1/2%3C223%3A%3AAID-RSA14%3E3.0.CO%3B2-O`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291098-2418%28199608/09%299%3A1/2%3C223%3A%3AAID-RSA14%3E3.0.CO%3B2-O`.