An Algorithmic Application of Cake-Cutting

Vladimir Fomin
Mathematics in Politics and Law
Dr. Philip Grech

January 26, 2018

Abstract

This paper deals with two algorithms which provide an envy-free solution to the three player Cake-Cutting Problem. Since Cake-Cutting is very useful in real world applications as well as widely applicable, we will try to implement the Stromquist Moving-Knife and the Selfridge-Conway procedure while still preserving their envy-freeness property. Afterwards, we will analyze both implementations numerically and compare them.

Contents

1 Introduction

The goal of this paper is to implement the Stromquist Moving-Knife as well as the Selfridge-Conway procedure in Matlab. First, we will need to introduce a mathematical model on which we want to apply these algorithms. Then we will briefly study Stromquist's method and compare it to the Selfridge-Conway procedure. After that we will describe some numerical methods to analyze the respective implementations. Lastly, we will solve the question if it is indeed possible to preserve envy-freeness while discretizing these algorithms.

1.1 The mathematical model

First, we have to define the Cake Cutting problem.

Definition 1 (Cake-Cutting Problem) *We call the set $\mathcal{P}\{C, (p_i)_{i=1}^n\}$ a n-player Cake Cutting Problem.*

i) C is some interval on $\mathbb{R}$ referring to the cake.

ii) n is the amount of players involved.

iii) $p_i : C \to \mathbb{R}_+$ *is the continuous and integrable preference function of each player with the following property:* $\forall i \in \{1, \ldots, n\} : \int_C p_i(x)dx = 1$.

From now on we will implicitly assume C to be the unit interval and n to equal 3, since the Stromquist as well as the Selfridge-Conway procedures only work with 3 players.

Next we define a solution to the Cake-Cutting Problem.

Definition 2 (Solution of the Cake-Cutting Problem) *A solution to the Cake Cutting Problem $\mathcal{P}$ is a function $\mathcal{S}$ with the following property:*

$$\mathcal{S}(\mathcal{P}) = (I_i)_{i=1}^n \text{ s.t. } \bigcup_{i=1}^n I_i = [0,1]$$

where $I_i \subset \mathbb{R}$ is a collection of subintervals of $[0,1]$ assigned to player i. We call the solution proportional if it holds:

$$\forall i \in \{1, \ldots, n\} : \int_{I_i} p_i(x)dx \geq 1/n.$$

Additionally, if the following holds:

$$\forall i, j \in \{1, \ldots, n\} : \int_{I_i} p_i(x)dx \geq \int_{I_j} p_i(x)dx,$$

we call the solution envy-free.

Remark 3 *As we have seen in the lecture, any envy-free solution is also proportional. But proportionality does not always imply envy-freeness.*

2 The Stromquist Moving-Knife Algorithm

In this section we will cover the Stromquist Moving-knife [1] algorithm and prove that its solution is envy-free. Consider the following setup. There are three players with their respective preference functions $p_1(x)$, $p_2(x)$ and $p_3(x)$, as well as a referee.

- The referee moves a sword in a continuous manner from the left to the right part of the cake. The sword divides the cake into two parts.
- Each player holds a knife over the right part of the cake such that each knife divides that part into two equal pieces according to their respective preference functions.
- At each point in time, there is a leftmost, a middle and a rightmost knife. Thus, the cake is divided into three pieces by the sword and the middle knife.
- If some player shouts "cut" at any point in time, that player receives the part left of the sword. The middle piece is given to the remaining player whose knife is closest to the sword. The rightmost piece is given to the last remaining player.

So when does it make sense for any player to shout cut? The following proposition provides an answer.

Proposition 4 *Every player can make sure that the piece they receive is at least as big as any other piece according to their preference function. Each player shouts "cut" as soon as the leftmost piece is equal in value to the piece they would receive if they remained quiet and some other player shouts "cut".*

Proof Note that in the eyes of the player who shouts and receives the leftmost piece, that piece has the same value as the piece their knife rests on and thus is at least as high in value as the remaining piece. Hence, the solution is envy-free to the player who shouts.

It remains to check if the solution is also envy-free according to both players who remaining quite.

i) Suppose the player with the leftmost knife remains quiet. They would receive the middle piece, which to them is larger in value than the leftmost piece since they remained quiet. Also, their knife rests on the middle piece and hence their piece is at least as big as the rightmost piece.

ii) A symmetric argument holds for the player with the rightmost knife.

iii) The player with the middle knife divides the two pieces to the right equally. By remaining quiet, they receive either the rightmost or the middle piece and therefore the solution is also envy-free according to the middle player.

■

2.1 A Comparison to Selfridge-Conway

In the lecture, we have already seen the Selfridge-Conway [2] procedure which also provides an envy-free solution to the Cake-Cutting problem. What are the key differences?

Stromquist	**Selfridge-Conway**
• 3 players	• 3 players
• Envy-free	• Envy-free
• Always 2 cuts	• At most 5 cuts
• Connected pieces	• Disconnected pieces
• Moving-Knife protocol	• Discrete protocol

By comparing both protocols we immediatly see that the Stromquist's algorithm is superior in terms of properties. By using the method, one obtains connected pieces with a minimal number of cuts. This provides a huge benefit in real world applications if the resource being distributed is difficult to disconnect.

The last bullet point may pose a problem though. Intuitively, a discrete type protocol should be easier for a computer to work with than a continuous one. What benefit is any algorithm if it is too costly to execute? This question will be addressed in the conclusion.

3 Discretization

In chapter one, we defined the underlying mathematical model. The next step is to take that model and transform in such a way that we can feed it into a computer. The simplest way to analyze any algorithm numerically is to observe its behavior regarding many different scenarios. The goal of this chapter is to provide some intuition of how to generate these problem scenarios $(\mathcal{P}_i)_{i=1}^n$ in a "reasonable" way.

3.1 Generating Problem Instances

Our aim is to generate Cake-Cutting problem instances such that they properly reflect the underlying mathematical model. But this means that we need to consider any possible continuous function. At first, this seems to be a hopeless endeavour, but help comes in the form of the famous approximation theorem by Weierstrass.

Theorem 5 (Weierstrass Approximation Theorem) *Suppose $a < b \in \mathbb{R}$ and $f : [a, b] \to \mathbb{R}$ a continuous function. Then for every $\epsilon > 0$ there exists a polynomial p such that the following holds:*

$$\forall x \in [a, b] : |f(x) - p(x)| < \epsilon.$$

Essentially, this theorem states that we can approximate any preference function from our model arbitrarily well by polynomials. Hence, we only need to find a way to generate such polynomials. The following procedure does the trick.

Generating Polynomial Preference Functions

- Define the sample size m as well as the number of interpolation points k.
- For each m divide the interval $[0, 1]$ into k equally sized subintervals.
- Assign random numbers in $[0, 1]$ to each border point of these subintervals.
- For each m interpolate the given points according to spline interpolation. [3] [4]
- Find the minimal value of each $spline_m$ and shift it upwards if it is below 0.
- Divide each $spline_m$ by its integral from 0 to 1.

Remark 6 *Note that we could have interpolated the points directly by a polynomial of sufficient degree, but splines have desirable numerical properties, such as more stable behaviour towards 0 and 1.*

Intuitively, we can capture the behaviour of a wide variety of preference functions by adjusting the sample size as well as the number of interpolation points. By generating a large enough sample and partitiong it into sets of three we can generate an arbitrary amount of problem instances. One such problem instance along with its solutions can be seen in Figure 1.

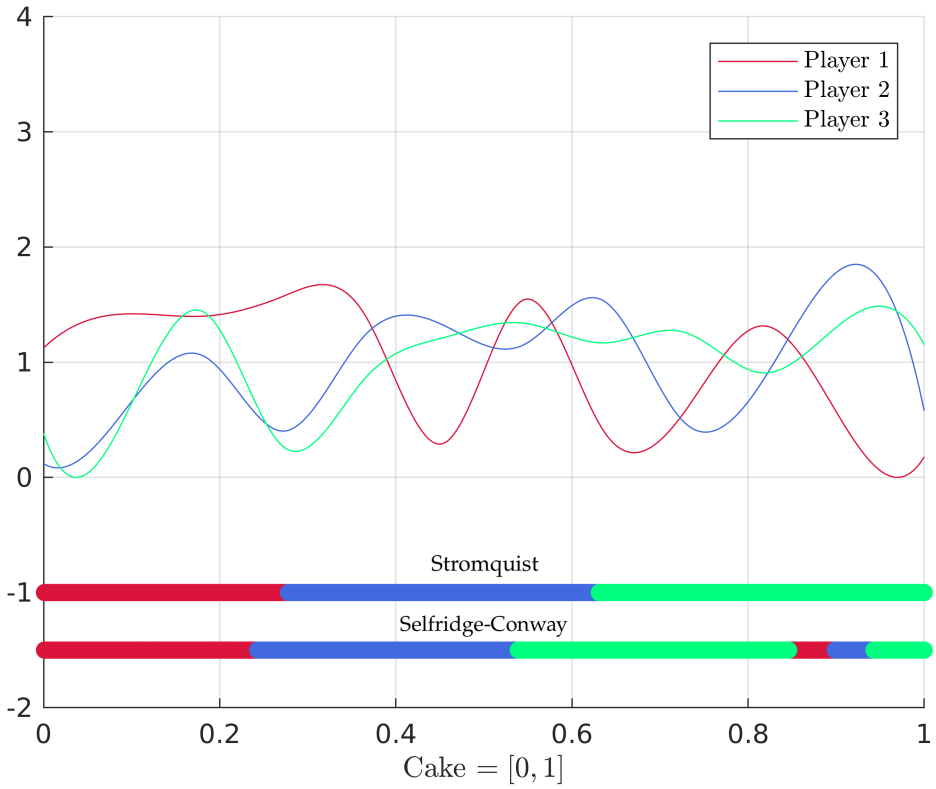

Figure 1

Additionally, by generating a huge number of splines and plotting their average we can see that the average is close to the uniform preference function (Figure 2). In some sense, this means that we are able to capture the average case in a reasonable way.

3.2 Preserving Envy-Freeness

Now we are able to analyze both implementations regarding the average case. We compute the solution to a large number of instances and study the properties of the solutions. Since both algorithms provide an envy-free solution in the theoretical setting, we are particularly interested if the implementations preserve that property.

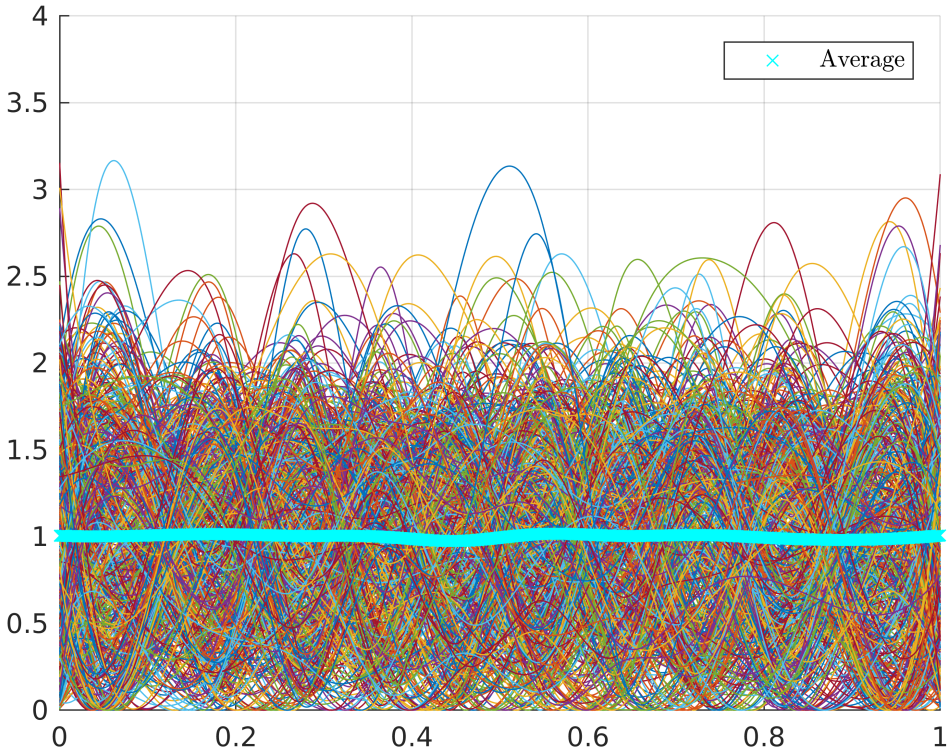

Figure 2

Dissapointingly, the answer is somtimes. The tricky part is that we have to compute the value of some piece. This computation is done via numerical quadrature and leads to follow up errors when assigning the pieces $\mathcal{S}(\mathcal{P}) = (I_i)_{i=1}^n$ later on. But as we see in the next sections that answer does not provide the whole picture.

3.2.1 Frequency of Envy-Free Solutions

As mentioned the implementations preserve envy-freeness only sometimes. Then the natural follow up question is: How often do they preserve envy-freeness? We plot the frequency of envy-free solutions of the Stromquist as well as the Selfridge-Conway procedure depending on the number of quadrature steps for 200 distinct problem isntances. This leads to Figure 3. The frequency of the Stromquist procedure stops increasing after some threshold of λ at around 78%. Though the Selfridge-Conway seems to keep increasing. Sadly we are not able to study it further due to computation time.

3.2.2 Relative Error of Non-Envy-Free Solutions

The last question which we want to consider is: If some computed solution is not envy-free, what is its relative error? First, we need to define what we mean by the relative error.

Definition 7 (Relative Error) *The relative error of a computed solution $\hat{\mathcal{S}}$ is the quantity:*

$$e(\hat{\mathcal{S}}) = \max_{\substack{i\in\{1,2,3\} \\ j\in\{1,2,3\}-\{i\}}} \left\{ \frac{|\int_{I_i} p_i(x)dx - \int_{I_j} p_i(x)dx|}{\int_{I_i} p_i(x)dx} \right\}$$

We plot the maximum relative error across all 200 problem instances once again depending on the number of quadrature steps λ. This leads to Figure 4. This time the Stromquist procedure behaves more nicely. Not only is the relative error smaller for all λ, but the behaviour seems to be more stable.

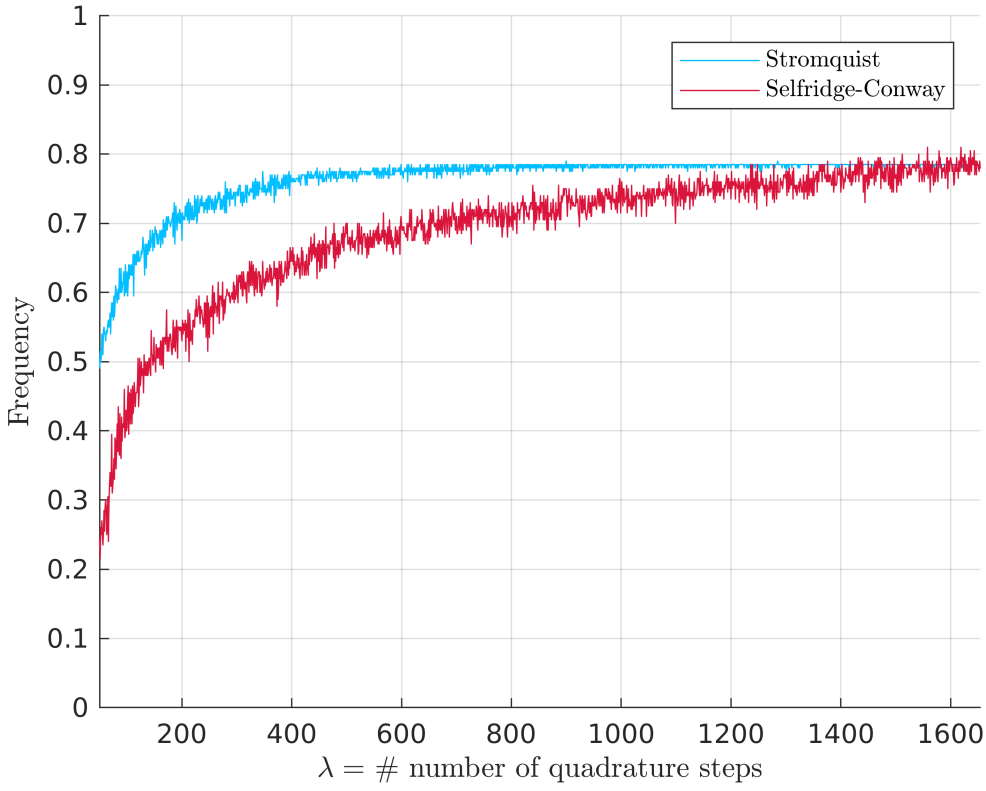

Figure 3

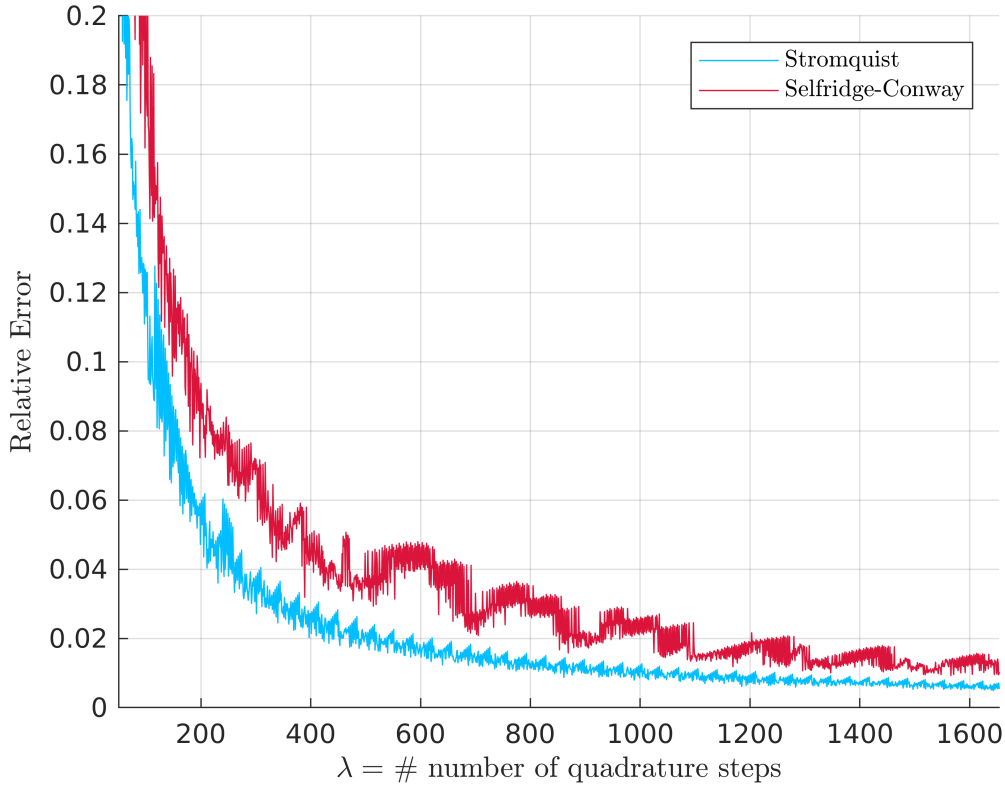

Figure 4

4 Conclusion

As seen in the figures, the implementations behave in a very nice way if we pick the quadrature step size large enough. Granted, this methodology has its flaws in that we did not prove the actual correctness of these implementations, but nevertheless the data shows that we get a good result most of the time.

Furthermore, implementing the Stromquist Moving-Knife procedure was not as difficult as originally expected. As one can mimic the nature of continuous protocols by iterating over small step sizes, the Stromquist Moving-Knife was indeed very well behaved. On the other hand, the Selfridge-Conway was a total mess requiring a number of IF-conditions. This coupled with the fact that the Stromquist Moving-Knife has superiour properties as well as better numerical behaviour implies that it is clearly the better choice for actual applications.

Also note that the running time was not included since both implementations lack the necessary optimization. Both algorithms should require around the same number of operations anyway as each section of the cake needs to be evaluated by the preference functions at least once by both algorithms.

On a personal note, I really enjoyed writing this paper as it proved to be a valuable learning experience as well as provided me as well as every interested reader with algorithms which one can actually use. Though, cramming this much information into that tight of a space was a considerable challenge.

Remark 8 (The Actual Matlab Source Code) *I published the source code of both algorithms according to the GPL-License. It can be downloaded or copied from the Github project in the references [6].*

i) To generate similar plots simply execute the script named "execute_me.m". Explanations are provided in the form of comments.

ii) Functions that deal with interpolation and integration contain the prefix "ip_". Similarly, functions that perform various calculations regarding the Stromquist Moving-Knife and Selfridge-Conway are prefixed "mk_" or "sc_" respectively.

iii) Creating the figures regarding the frequency and the relative error took around 4 days of computational time. They will not be included in the script "execute_me.m".

References

[1] The Stromquist Moving-Knife Procedure,
The American Mathematical Monthly Vol. 87, No. 8 (Oct., 1980), pp. 640-644

[2] Cake-Cutting Algorithms: Be Fair If You Can,
Webb, William (1998). Natick, Massachusetts: A. K. Peters

[3] Cubic-Spline Interpolation,
Paper by Mckinley and Levine

[4] Spline Interpolation
Hazewinkel, Michiel, ed. (2001) [1994], Encyclopedia of Mathematics, Springer Science+Business Media B.V. / Kluwer Academic Publishers

[5] Average-Case Analysis of Numerical Problems,
Ritter (2000), Springer-Verlag Berlin Heidelberg

[6] Matlab Code : https://github.com/fominv/Stromquist-Moving-Knife-and-Selfridge-Conway