A Counterexample to the Core Protocol of Aziz and Mackenzie

Christian Bechler and Kenan Bitikofer

Goshen College

July 2017

Abstract

In 2016, Dr. Haris Aziz and Simon Mackenzie [1] proposed an algorithm that gives an envy-free (EF) division of a cake in time bounded by the number of agents with an upper bound of $n^{n^{n^{n^n}}}$. The workhorse of this algorithm is the Core and Subcore protocols, which give a partial EF allocation of the cake and a bonus value of the cutter over some other agent(s). Ultimately, we discovered edge-case errors in Aziz and Mackenzie's SubCore protocol through implementation and random testing, which we were able to trace back to errors in their proof of correctness for the Core protocol. We give in this paper a succinct counterexample to demonstrate the error, and pinpoint the faulty logic in their proof.

1 Introduction

Since its introduction in 1948, finding an algorithmic method for envy-free cake-division has intrigued mathematicians, computer scientist and social scientists alike. The goal is to completely divide up a cake C (i.e. any divisible, uni-dimensional, heterogeneous good) among some set of agents A such that no agent wants to trade their piece with another agent's piece (i.e. envy-freeness, which we will subsequently abbreviate as EF).

For two agents, such an algorithm is easy to understand: one agent divides the cake into two equal slices and the other chooses. Three agents is slightly more challenging, although a solution was provided in 1960 by Selfridge and later independently discovered by Conway which provided an EF allocation of the entire cake for three agents. However, a general solution remained out of reach for another three decades. In 1995, Brams and Taylor [2] provided an algorithm which finished in finite time, but could require an arbitrarily large number of Robertson-Webb [3] queries before it finished. An unbounded, general solution continued to evade researchers.

In 2016, Aziz and Mackenzie [1] proposed a discrete algorithm which purportedly returns an EF allocation in a bounded number of Robertson-Webb queries: $n^{n^{n^{n^n}}}$, where n is the

number of agents. This algorithm has been preliminarily accepted as the first and only algorithm to accomplish these goals. Critical to the success of this algorithm is the Core and SubCore protocols, which in conjunction, are purported to provide a partial, EF allocation that guarantees one agent a positive advantage over another.

We will demonstrate that the current SubCore Protocol will not run to completion given certain agent preferences, and therefore Core will not always return a partial EF allocation. Further, we will show where the proofs by Aziz and Mackenzie regarding the correctness of Core and Subcore are flawed. Finally, we will examine the possible implications of this discovery on the open problem of establishing the boundedness of EF cake cutting.

1.1 Model

Following Aziz and Mackenzie [1], we model a cake as an interval $[0, 1]$ and a piece of cake is any finite union of disjoint subintervals of $[0, 1]$. Similarly, any agent $a_i \in A = \{a_1, a_2, ..., a_n\}$ has a valuation function V_i such that for any $k_1, k_2 \in [0, 1]$ satisfying $k_1 \leq k_2$ $V_i([k_1, k_2]) \geq 0$ and $V_i([k_1, k_1]) = 0$. Furthermore for any two disjoint intervals $[k_1, k_2], [k_3, k_4] \subseteq [0, 1]$ the value $V_i([k_1, k_2] \cup [k_3, k_4]) = V_i([k_1, k_2]) + V_i([k_1, k_2])$. Finally, for any interval $X \in [0, 1]$ and $0 \leq \lambda \leq 1$ there exists some $X' \subseteq X$ such that $V_i(X') \leq \lambda V_i(X)$.

1.2 Core and SubCore Protocols

The Core protocol is proclaimed to be the only known protocol for providing a proportional, partial, EF allocation, which was also an open problem before Aziz and Mackenzie's work. Core has a specified agent divide the cake into n pieces of equal value where n is the number of agents. Then SubCore is called on the remaining agents and the n pieces. The SubCore call within Core is expected to return a neat allocation.

Definition 1.1. Taken verbatim from [1]
Consider a cake divided into n pieces and $m < n$ agents. An allocation of the cake into m agents is *neat* if (i) each agent's allocation is a (not necessarily whole) part of one of the pieces; (ii) one piece is unallocated; (iii) no agent prefers an unallocated piece over his allocation; and (iv) no agent prefers another agent's allocation over his allocation.

Note that by (i), (ii) should read "*at least* one piece is unallocated." A neat allocation allows for the cutter, who was not passed into subcore, to receive one of their originally cut pieces. This results in a partial EF allocation which is proportional from the cutter's perspective.

In Algorithms 1 and 2 we have used the Core and Subcore protocol pseudocode nearly verbatim from Aziz and Mackenzie's paper.[1] We have edited out certain lines which compute values used only in their proof and not in the operation of the algorithm. Our modifications are surrounded with square brackets to indicate the change.

Algorithm 1 Core Protocol

Input: Specified cutter (say agent $i \in N$), agent set N such that $i \in N$, and unallocated cake R.

Output: An envy-free allocation of cake $R' \subset R$ for agents in N and updated unallocated cake $R \setminus R'$.

1: Ask agent i to cut the cake R into n equally preferred pieces.

2: Run SubCore Protocol on the n pieces with agents set $N \setminus \{i\}$ [...]. The call gives an allocation to the agents in $N \setminus \{i\}$ such that one of the n pieces is untrimmed and unallocated.

3: Give i one of the unallocated untrimmed pieces from the previous step.

4: **return** envy-free partial allocation (in which each agent gets a connected piece) as well as the unallocated cake.

Algorithm 2 SubCore Protocol

Input: Cake cut into n pieces to be allocated among agents in set $N' \subset N$ with $n' = |N'|$.

Output: A neat envy-free partial allocation for agents in N' in which each agent $j \in N'$ gets a connected piece.

[Storing maximal piece value]

Order the agents in N' lexicographically. Suppose the order of agents is $1, 2, 3, ..., n'$. The order must be consistent with previous calls of the SubCore protocol.

Ask each agent to rank the pieces according to their valuations over the pieces. If an agent is indifferent among some pieces, we break ties based on the ranking he gave in the SubCore calling this SubCore. If this is the root SubCore, the ties are broken lexicographically. The ranking is used for tie-breaking between pieces if an agent derives equal value from two different pieces

for $m = 1$ to n' **do**

  if there is an unallocated piece that gives agent m the highest value among all pieces **then**

    we tentatively give agent m that piece and go to the next iteration of the 'for' loop. {If m has multiple most preferred unallocated pieces, we tentatively give him the most preferred unallocated piece ranked highest by the agent}

  else

    — the first m agents are contesting for the same $m - 1$ tentatively allocated pieces. We call them the *contested* pieces.

    — Then each agent in $[m]$ is asked to place a trim on all *contested* pieces of high enough value so that the contested piece on the right hand side of his trim is of the same value as his most preferred piece outside the $m - 1$ contested pieces.

    — [Benchmark value setting]

    Set W to be the set of agents who trimmed most (had the rightmost trim) in some piece. If a piece has multiple agents having a rightmost trim due to ties, then among those agents we place the agent with the lowest index in W.

    while $|W| < m - 1$ **do**

      Ignore the previous trims of agents in W from now on and forget the previous allocation.

      Run SubCore Protocol on the contested pieces with W as the target set of agents and for each contested piece, the left side of the right-most trim by an agent in $[m] \setminus W$ is ignored. {The result of the recursive call of SubCore is an allocation that gives a (partial) contested piece to each of the agents in W.}

      Take any unallocated contested piece a. The current left margin ([before] which the piece is ignored) is by agent $i \in [m] \setminus W$. In case there are multiple such agents, choose the agent with the lowest index. Give a to i.

$$W \leftarrow W \cup \{i\}.$$

    At this point the current allocation of agents in W is tentative and not permanently made. {An agent from $[m] \setminus W$ has been added to W. For the updated W, each agent in W gets a partial contested piece and no agent envies an unallocated piece. Recall that for each piece, the left side of the right-most trim by an agent in $[m] \setminus W$ is ignored.}

    end while

    Run SubCore on all agents in W and the set of contested pieces that are only considered [beyond] the trim of the agent in $[m] \setminus W$. [The benchmark is at least b_j]. {At this point $|W| = m - 1$ and each agent in W has a tentatively allocated contested piece}

    The only agent j remaining in $[m] \setminus W$ is tentatively given his most preferred uncontested piece. {If there are multiple most preferred uncontested pieces, we tentatively give him the most preferred uncontested piece for which he [has] the highest ranking.}

  end if

  [Update agent benchmarks]

end for

return envy-free partial cake for agents in N' (such that each agent gets a connected piece that is on the right hand side of the original piece he trimmed most) as well as the unallocated cake.

2 Counterexample to Aziz and Mackenzie's Subcore Protocol

We now will proceed to walk through a counterexample with 5 agents, whose preferences are constructed so as to break the operations of the SubCore protocol. The example is illustrated succinctly in Figure 1.

We begin with 5 agents, who we will call $\{a_0, a_1, a_2, a_3, a_4\}$, and a cake $[0, 1]$ to divide. The preferences of each agent a_i over the cake are given below, where the jth element of each 13-tuple V_i indicates how much a_i values the interval $\left[\frac{j-1}{13}, \frac{j}{13}\right]$. Note that, because different agents' preferences are never directly compared, normalization of the values is unnecessary.

$$V_0 = (1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 6, 6)$$
$$V_1 = (4, 4, 2, 1, 1, 8, 0, 0, 0, 0, 0, 8, 0)$$
$$V_2 = (1, 1, 1, 1, 1, 1, 1, 1, 3, 0, 0, 4, 0)$$
$$V_3 = (1, 1, 1, 1, 2, 2, 4, 2, 4, 2, 4, 4, 0)$$
$$V_4 = (1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 3, 0)$$

Core begins with a_0 cutting the cake into 5 pieces that she values equally:

$$\left[0, \frac{6}{13}\right], \left[\frac{6}{13}, \frac{9}{13}\right], \left[\frac{9}{13}, \frac{11}{13}\right], \left[\frac{11}{13}, \frac{12}{13}\right], \left[\frac{12}{13}, \frac{13}{13}\right].$$

Call the set of pieces, from left to right, $P = \{p_1, p_2, p_3, p_4, p_5\}$. Those 5 pieces are then passed into SubCore along with agents $N' = \{a_1, a_2, a_3, a_4\}$. Let us name this call of SubCore $SubC$.

Within $SubC$, the agents are already in lexicographic order (step 2). We have each agent give a ranking of the pieces by value, with ties broken lexicographically (step 3). Given the agent preferences, this process will produce the following rankings R_i for each agent $a_i \in N'$:

$$R_1 = \{p_1, p_4, p_2, p_3, p_5\}$$
$$R_2 = \{p_1, p_2, p_4, p_3, p_5\}$$
$$R_3 = \{p_2, p_1, p_3, p_4, p_5\}$$
$$R_4 = \{p_1, p_4, p_2, p_3, p_5\}$$

We begin the for loop (step 4), with $m = 1$; so a_1 chooses p_1 as her most valued piece (steps 5 and 6). Now we increment m, such that $m = 2$ (step 4). Agent a_2 chooses the same piece, and they both place trims on it so that the piece to the right of each trim is equal in value to the maximal value of the uncontested pieces $\{p_2, p_3, p_4, p_5\}$ (step 7). Specifically, a_1 trims p_1 at $\frac{5}{13}$ to be of equal value to p_4, that is $V_1\left(\left[\frac{5}{13}, \frac{6}{13}\right]\right) = V_1(p_4) = 8$. Agent a_2 trims p_1 at $\frac{1}{13}$ to be of equal value to p_2 In our case, a_1's trim is to the right of a_2's trim. As a result a_1 is a winner and W = $\{a_1\}$ (step 8). Since $|W|$ is not less than $m - 1$ we skip

the while loop (step 9). Agent a_1's previously placed trim on p_1 is forgotten (Step 13), and SubCore is called recursively on $W = \{a_1\}$ and the set of contested pieces $\{p_1\}$ (step 14). Unsurprisingly a_1 chooses p_1 and is allocated everything to the right of a_2's trim. Returning to $SubC$, a_2 is the last non-winner and must choose from the uncontested pieces, selecting p_2 in accordance with her ranking (step 15).

Now we increment m to 3, and have a_3 choose her most preferred piece, which is p_2. Since p_2 is already allocated to a_2 all three agents in $[m]$ must trim the two contested pieces p_1 and p_2 (step 7). Since p_4 remains uncontested, a_1 trims p_1 to be of equal value to p_4 again, and does not trim p_2 at all (because $V_1(p_2) < V_1(p_4)$). a_2 trims both p_1 and p_2 to be of equal value to p_4 and a_3 trims both p_1 and p_2 to be of equal value to p_3. Because of their preferences a_2 and a_3 both trim p_1 and p_2 to the same point. The rightmost trim on p_1 belongs to a_1 and the rightmost trim on p_2 belongs to a_2 because a_2 and a_3 placed a trim at the same place on p_2 and ties are broken lexicographically. Thus $W = \{a_1, a_2\}$ (step 8). Since $|W| = m - 1$ we skip the while loop, forget the trims by a_1 and a_2, and run SubCore on W and the set of contested pieces $\{p_1, p_2\}$ to the right of the trims by a_3 (step 14). We name this call of SubCore $Sub2$. Since a_1 sees no value in p_2, a_1 will choose p_1 ($Sub2$, steps 5 and 6). Similarly, p_1 and p_2 are both delimited on the left by the trims of a_3, which limit the value of p_1 and p_2 to the value of p_4 from a_2's perspective. Thus, a_2 will choose p_2 because it is unallocated and of equal value to p_1 ($Sub2$, steps 5 and 6). We return from $Sub2$ to $SubC$ with these allocations. Now a_3 is the last non-winner and selects p_3 from the uncontested pieces in accordance with her ranking (step 15).

We increment m to 4 (step 4), and have a_4 choose her most preferred piece, which is p_1. Since p_1 is already allocated to a_1 all four agents in $[m]$ must trim the three contested pieces p_1, p_2 and p_3 (step 7). Again, a_1 trims p_1 at $\frac{5}{13}$ to be of equal value to the value of p_4, but does not trim p_2 or p_3 because she sees no value in those pieces. Similarly, a_2 again trims p_1 at $\frac{2}{13}$ and p_2 at $\frac{7}{13}$ to be of equal value to the value of p_4, but does not trim p_3 because she also sees no value in that piece. On the other hand, a_3 trims p_1 at $\frac{4}{13}$, p_2 at $\frac{8}{13}$, and p_3 at $\frac{10}{13}$ to be of equal value to p_4. Finally a_4 trims p_1 at $\frac{3}{13}$ to be of equal value to p_4, but does not trim p_2 or p_3 because she sees no value in them. Note that on p_1, agent a_1 is again the rightmost cutter. Also note that on pieces p_2 and p_3, agent a_3 is the rightmost cutter. Therefore $W = \{a_1, a_3\}$ (step 8). Because $|W| < m - 1$, we enter the while loop. We forget the trims made by a_1 and a_3 (step 10), and call SubCore recursively (we label the call $Sub1$) on $W = \{a_1, a_3\}$ and on the set of contested pieces $\{p_1, p_2, p_3\}$ to the right of the trims made by agents a_2 and a_4. In $Sub1$, agent a_1 will again choose p_1, as she sees no value in p_2 or p_3 ($Sub1$, steps 5 and 6). Now agent a_3 values the three pieces at 5, 6, and 6, respectfully. Since pieces p_2 and p_3 are equally of most value to her, agent a_3 chooses based on her ranking. The ranking $R_{3,Sub1} = \{p_2, p_1, p_3\}$, because the ranking in $Sub1$ has ties broken using the ranking from the call above, $SubC$. Thus, a_3 will choose p_2 ($Sub1$, steps 5 and 6). We return from $Sub1$ and proceed to the next step (step 12), adding an agent to winners by taking the new rightmost trimmer on an uncontested piece. However, the only uncontested piece is p_3, which has no trims, and thus there is no rightmost trimmer. Therefore SubCore breaks at this point (step 12).

Because SubCore is unable to complete the allocation, Core also cannot complete. As a result, Aziz and Mackenzie's algorithm will terminate without providing an EF allocation given these agent preferences.

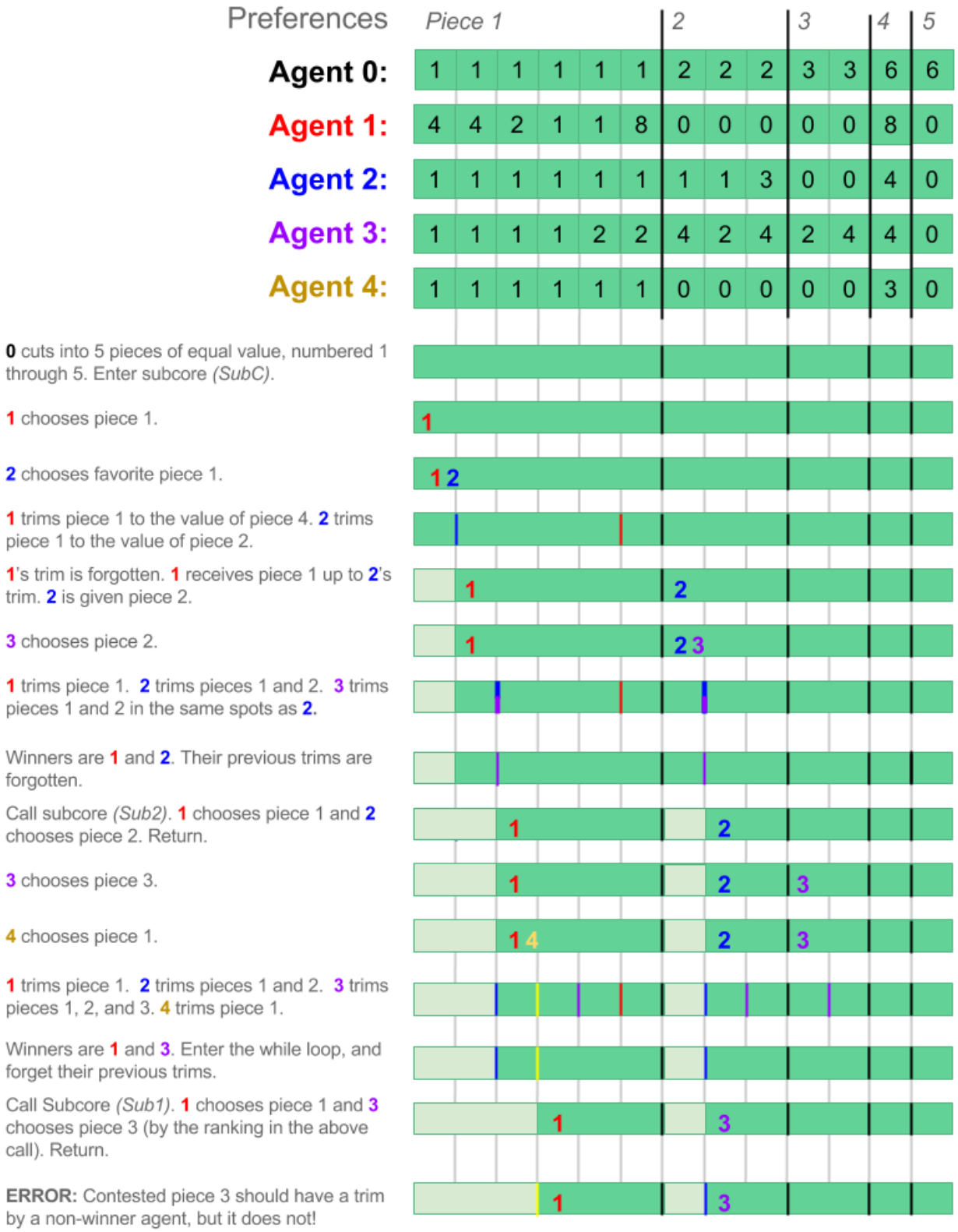

Figure 1: Illustrated run-through of the counterexample.

3 The Error in the Proof

Aziz and Mackenzie claim that such a situation described above is impossible in their proof of Lemma 4.5. While arguing that another agent can always be added to winners, they attempt to show that it is impossible to have an unallocated contested piece with no trims. They write:

> Case 2: Contested unallocated piece a has no marginal trim (on the current left margin) by a non-winner. Let us refer to the current call of the SubCore protocol as $SubC$. We want to show that Case 2 cannot happen when we call a new SubCore protocol referred to as $Sub1$ on a set of contested pieces C_1 and agents W_1. Let a be the contested piece satisfying the conditions of Case 2. Since a is in C_1 either a is one of the agents in W_1's most preferred piece or at some previous call of the SubCore protocol i was 'kicked out' from contested pieces and given piece a, thus making a a contested piece. Let us call this previous call $Sub2$. Note that just like $Sub1$ this call was made by SubC, which means that ties are broken in the same way between agents and pieces. Let us say that $Sub2$ was called on agents W_2 and contested pieces C_2 such that $a \notin C_2 \subset C_1$.[1]

In order to understand this we will consider our counterexample and the above proof together. Let agent i be a_3 and let piece a be p_3. We can claim this because p_3 has no trims on it and agent a_3 is agent i because they were 'kicked out' from the contested pieces and had to choose p_3 at an earlier SubCore call. We can label this earlier call of SubCore $Sub2$ just as Aziz and Mackenzie did because p_3 was not in the contested pieces at this call and a_3 was forced to take it. Likewise we can label the parent SubCore call of $Sub2$ $SubC$. Finally the recursive call of SubCore from within $SubC$ where a_3 should choose p_3 we will call $Sub1$ in line with our above proof. Aziz and Mackenzie go on to write:

> ...We will now argue that in $Sub1$ i must get piece a, showing that Case 2 cannot happen... For any remaining piece c both in C_1 and C_2, we will now show that i cannot get c. To show this, look at the agent j_0 who got c in Sub2. When we call Sub1, either j_0 is in W_1 or is not. If he is not, then he is a non-winner, and has placed a trim on c which must be to the right of or on the part of c he was allocated in $Sub2$. This follows from the fact that if j_0 is a non-winner he is asked to equalise c to his most preferred uncontested piece, will be worse than the part of c j_0 got in $Sub2$ (else neatness breaks). Agent i was not envious of the lesser trimmed c when allocated a, and if indifferent ties were broken in favour of a. [1]

At this point we must disagree with Aziz and Mackenzie. Rather than claim that in $Sub1$ i must get piece a, we will instead argue that i may not get piece a showing that Case 2 can happen. We will do this by demonstrating that i could get c. In this scenario let piece c be p_2 and have agent j_0 be a_2. We can claim this because in $Sub1$, a_3 chose p_2, just as agent i would choose piece c in $Sub1$ according to Aziz and Mackenzie. Similarly, we know in $Sub2$, a_2 chose p_2 just as agent j_0 would choose piece c in $Sub1$ in Aziz and Mackenzie's example. However, as we showed above a_3 can get c in our scenario. Here is where the proof breaks down, because our example clearly demonstrates that in $Sub1$ there can exist an agent i who

would take piece c rather than a piece without trims a. The reason why Aziz and Mackenzie come to the opposite conclusion is the following claim: "if indifferent, ties were broken in favour of a." Our above example demonstrates this is false because a_3 does not get to choose between p_2 and p_3 because a_3 is the last non-winner. Instead, line 14 of SubCore has a_3 choose p_3 from the uncontested pieces though the ranking would indicate that she prefers p_2. Therefore it is possible for an unallocated, contested piece without trims to exist. This, in turn, implies that it is not always possible to continue adding agents to winners as the Core Protocol stands.

4 Discussion

We have demonstrated that the Core and SubCore Protocols as proposed by Aziz and Mackenzie will not run to completion given certain agent preferences, and therefore Core will not always return a partial EF allocation. We have done this with a specific and simple example of agent preferences, and have shown where the proofs of Aziz and Mackenzie have overlooked a particular situation. This counterexample shows that Aziz and Mackenzie, or other researchers, have a bit of work yet to do, either in patching up this flaw in SubCore, or in discovering a different partial allocation algorithm to take Core's place in the larger algorithm.

The difficulty in this second option is that Aziz and Mackenzie's Main Protocol requires that Core returns a partial allocation where the cutter agent has a positive bonus value over some other agent or allocate the remainder of the cake. Waste Makes Haste, another recently devised partial EF algorithm that is simpler and less efficient than Core, does not guarantee such a bonus value.[4]

These questions supposedly settled by Aziz and Mackenzie are reopened, at least for the moment:

Does there exist a bounded algorithm that returns a proportional and EF partial allocation?

Does there exist a bounded algorithm that guarantees completeness and envy-freeness for any number of agents?

Acknowledgements

This research was supported by the Goshen College Maple Scholars program and the Goshen College Math Department. Special thanks to Dr. David Housman, who has given essential advising and provided direction to the research.

References

[1] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–427. IEEE, 2016.

[2] S. J. Brams and A. D. Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

[3] J. M. Robertson and W. A. Webb. *Cake Cutting Algorithms: Be Fair If You Can.* A. K. Peters, 1998.

[4] Erel Segal-Halevi, Avinatan Hassidim, and Yonatan Aumann. Waste makes haste: bounded time algorithms for envy-free cake cutting with free disposal. *ACM Transactions on Algorithms (TALG)*, 13(1):901–908, 2016.

The code for our python implementation of Core and SubCore is freely available at http://github.com/kenanbit/envy-free-cake