COMS4444 Project 3 Report

Team 5: Emma Corbett, Fenglei Gu, Justin Hudis

November 15, 2023

Contents

1 Introduction **3**

2 Initial Design **4**

2.1 Uniformity U=0 4

2.2 Optimal Solution for $n = 2$ 5

2.3 Default Solution for $n = 3$ 6

2.4 Intuitive Solution for $n = 4$ 6

3 Intermediate Design **8**

3.1 $n = 2$: Moving Cut Center Inward 8

3.2 $n = 3$: An Extension of Our $n = 2$ Solution 8

3.3 $n = 4$: An Extension of Our $n = 3$ Solution 10

4 Final Design **11**

4.1 Taking Advantage of Symmetry 11

4.2 Permuting Pizza Toppings Based on Preference Distribution 12

5 Tournament Results **16**

6 Discussion **22**

6.1 Conclusions 22

6.2 Future Work 22

7 Summary of Contributions **23**

8 Acknowledgements **23**

1 Introduction

This project presents an interesting problem of pizza making and cutting. In a normal pizza restaurant:

- The pizza toppings are distributed (almost uniformly) randomly on the pizza.
- The pizza is often cut into several (in this project, 8) pieces by diameter lines with the intersecting point of the cuts located at the center of the pizza circle.

In this project, however, we need to design a special type of pizza, such that:

- When prompted by the couple (customers), the pizza should be cut into 8 pieces with 4 cuts 45 degree to each other - in the same way as stated by the **Pizza Theorem** - and each person should get a better distribution of toppings based on their preferences.
- When not prompted, the pizza should be cut to 8 pieces in an ordinary way (called "**central cut**") and each person should get a uniform distribution of toppings.

In this project, there are three subtasks - in each subtask, there can be 2, 3, or 4 toppings on a pizza. For each subtask, there are two measures:

- Uniformity U: measuring how uniform the portion of topping that two customer gets when they do not order special service. It is calculated by L1 loss, i.e., sum of absolute value of difference between ideal uniform amount of each topping and the real amount of each topping that a customer gets.
- Satisfaction S: measuring how much improvement that our special cut has made catering for customer's preference by comparison with a uniform distribution. To calculate S, we first calculate B, the L1 loss (as defined above) of baseline cut (uniform distribution of toppings for each customer); then calculate C, the L1 loss of our specialized cut; and $S = C - B$.

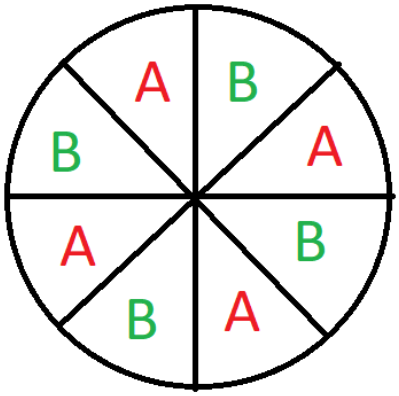

Figure 1: Initial prototype of pizza layout design for optimal uniformity $U = 0$.

2 Initial Design

2.1 Uniformity U=0

To begin with, we first aim to design a layout such that the uniformity is optimized. To do this, we came up the following prototype (see Figure 1): We first evenly divide pizza to 8 slices of $\frac{\pi}{4}$. Then, in each slice, we fill uniformly (in angular dimension) with one topping.

Let $a_i, b_j (i, j \in \{0, 1, 2, 3\})$ be $n (n \in \{2, 3, 4\})$ dimensional vector representing the amount of each topping (in the subtask with n toppings) in each slice. When $a_0 + a_1 + a_2 + a_3 = b_0 + b_1 + b_2 + b_3$, and we give the a_i slices to Alice, b_j slices to Bob, both Alice and Bob gets $a_0 + a_1 + a_2 + a_3 = b_0 + b_1 + b_2 + b_3$, i.e., uniform distribution of toppings. When the central cut is rotated by θ (without loss of generality, we can assume $\theta \in [0, \frac{\pi}{4})$; if $\theta \in [\frac{\pi}{4}, \frac{\pi}{2})$, it's the same due to symmetry between Alice and Bob; otherwise, taking the modulo $\frac{\pi}{2}$ as the cuts is rotational symmetric), then Alice gets $(1 - \frac{4\theta}{\pi})(a_0 + a_1 + a_2 + a_3) + \frac{4\theta}{\pi}(b_0 + b_1 + b_2 + b_3) = (1 - \frac{4\theta}{\pi})(a_0 + a_1 + a_2 + a_3) + \frac{4\theta}{\pi}(a_0 + a_1 + a_2 + a_3) = a_0 + a_1 + a_2 + a_3$, i.e., Alice and Bob still gets uniform distribution of toppings. Therefore, $U = 0$ for any central cut in this layout.

As we will see later, the restrictions set above are sufficient but not necessary for $U = 0$.

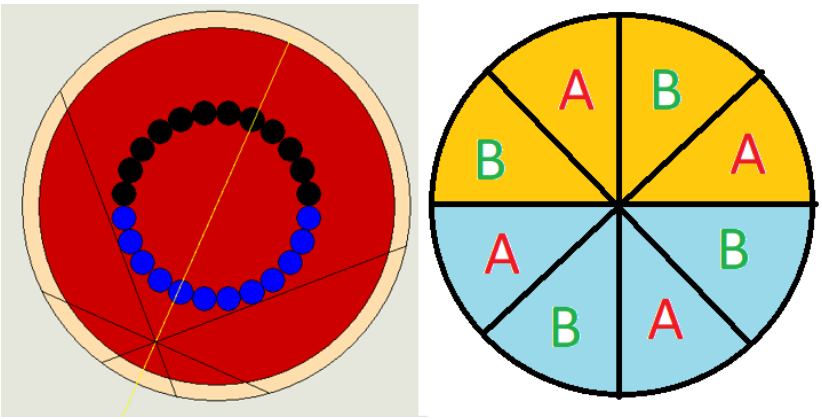

Figure 2: Initial pizza layout for $n = 2$.

2.2 Optimal Solution for $n = 2$

When $n = 2$, one way to put the toppings is as in Figure 2: We put one topping on the top semi-circle, and the other topping on the bottom semi-circle. Noticing that we only need uniformity in angular dimension, we could put toppings on arcs (center is at the center of the pizza) for simplicity in design, as seen in the figure.

Now we cut the pizza such that exactly one cut (the yellow one as depicted in the figure) would cut through the pizza center and the arcs (inner-circle); and that the off-center (of the cut) is chosen such that two other cuts are tangent to the toppings. By choosing different angle for the yellow cut, we could give Alice any proportions $(x, 12 - x) \forall x \in [0, 12]$; in this way, we would give Bob $(12 - x, x)$ toppings.

Now we prove that we can always minimize their unsatisfaction. Let's say Alice and Bob want $(a, 12 - a)$ and $(b, 12 - b)$ respectively. We need to minimize

$$L := |a - x| + |(12 - a) - (12 - x)| + |b - (12 - x)| + |(12 - b) - x| = 2(|x - a| + |x - (12 - b)|)$$

It's easy to see that L is minimized when x takes a value such that $(x - a)(x - (12 - b)) \leq 0$ (i.e., x takes any value between a and $12 - b$). Since both a and $12 - b$ takes value between 0 and 12, and x can take any value between 0 and 12, we can always pick an x that minimizes unsatisfaction L.

While there is a range that all minimizes unsatisfaction, we observe that catering for any customer's preferences (i.e., taking $x = a$ if prioritizing Alice, or $x = 12 - b$ if prioritizing Bob) would minimize the unsatisfaction as well (actually these are the boundries for the range). Hence, we decided to just prioritize Alice's preferences to make implementation easier.

2.3 Default Solution for $n = 3$

In our initial layout design (Figure 1), there're 8 slots, each of which we can fill a topping into. However, 3 does not divide 8. So we couldn't fulfill $n = 3$ subtask with this design. In the initial stage, we didn't realize we could step out from the prototype above. Hence we chose to adopt the default solution (i.e., random player).

2.4 Intuitive Solution for $n = 4$

We know that people have always have their most liked and least liked toppings. Also, from discussions for $n = 2$, we get the intuition that if one customer's preferences are fulfilled, then S is optimized or close to optimized. Since Alice and Bob are symmetric, below we prioritize Alice. Depending on what Alice liked-disliked most, there can be 4 choose 2 = 6 pairs of liked-disliked pairs. For each pair, we pick one type of pizza from a column below (see Figure 3). In each column, there are two choices: Model 1 or Model 2. In case that Alice has a strong preference, such that she wants 50% or more of the most liked topping and nearly 0 of the least liked topping, we would choose Model 1, which gives $(0.5, x, 0.5 - x, 0) \forall x \in [0, 0.5]$ distribution of toppings. Otherwise, it's guaranteed that Alice wants at least 0.25 of the most liked topping and at most 0.25 of the least liked topping, so we would choose Model 2, which gives $(0.5 - x, 0.25, 0.25, x) \forall x \in [0, 0.25]$ distribution of toppings.

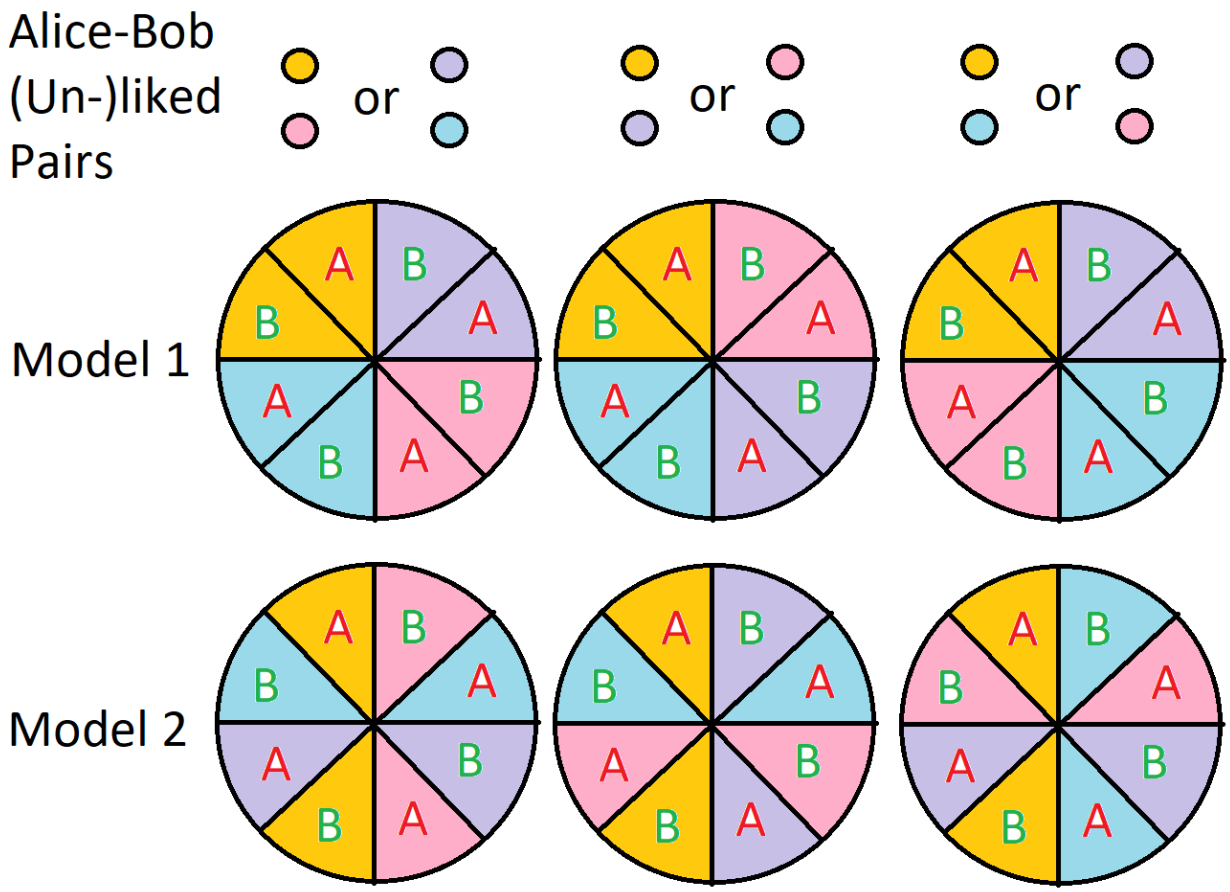

Figure 3: Initial layout design for $n = 4$. Like in $n = 2$ case, toppings are still put in arcs that form the inner circle, and we have exactly one cut that cuts through the inner circle, with the other two cuts tangent to the inner circle. (Not shown in this figure)

3 Intermediate Design

3.1 $n = 2$: Moving Cut Center Inward

Since our solution for $n = 2$ is optimal for metric S and U, there was not much room for improvement in the intermediate stage. However, Prof. Ross added a metric for cut distance from center, which teams were told to ideally minimize. To reduce this metric while maintaining our original optimality, we decided to use two concentric rings of toppings instead of one, and push the cut inward so that it is tangent with the outer circle, as seen in Figure 4.

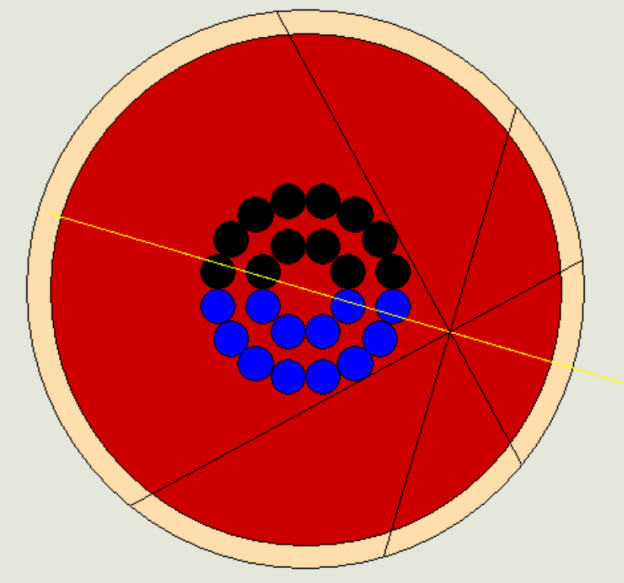

Figure 4: Using concentric circles to move the cut inward for $n = 2$.

3.2 $n = 3$: An Extension of Our $n = 2$ Solution

Our next goal was to extend our $n = 2$ solution to one for $n = 3$. Our TA Akshay pointed out that one could apply the $n = 2$ topping layout to two of the toppings and place the last topping in a blob in the corner. Doing so allows one to take advantage of the degree of freedom that is moving the cut inward and outward. As one does so, the topping in the blob will be apportioned differently to the two customers, assuming the blob intersects the

cut. We ran with this idea, although instead of a blob we settled on a 90-degree arc since that preserves $U = 0$. Since we restrict our cut centers to the top half of the pizza (each half provides identical options), we placed the arc above the inner toppings to ensure that our cut intersects the arc. The radius of the arc is such that the toppings are all touching exactly. This is an arbitrary decision that could be optimized with more time for investigation. In this stage, we always placed the first two toppings in the middle and the last one on the outside. That is, all of our pizzas looked like Figure 5. It is also worth noting that the inner topping layout looks a bit different from $n = 2$ due to the fact that there are fewer of each topping. We still kept the concentric circles, though, since doing so extends the range of potential cut radii further inward.

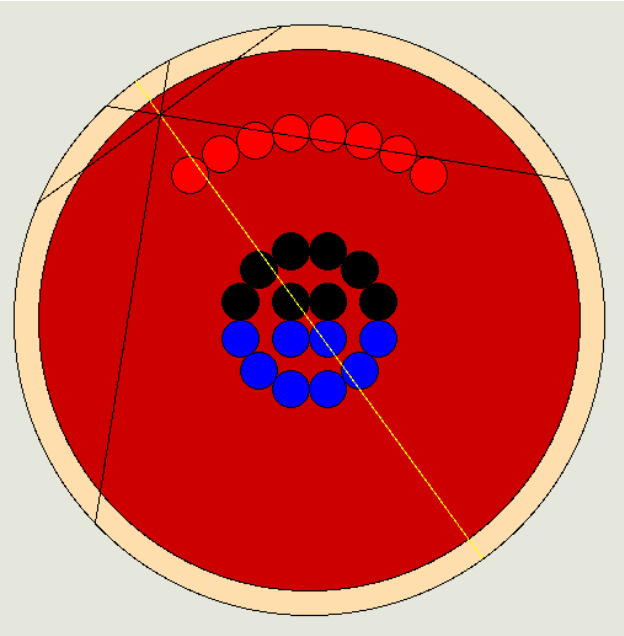

Figure 5: Topping layout and example cut for $n = 3$.

To determine where to make the cut, we apply a somewhat intelligent brute-force algorithm. It is likely possible to analytically determine an optimal cut location, but doing so gets challenging once there are more than 2 toppings. Instead, we take two steps: 1) fix radius at 5 and find the cut angle between 180 and 360 which minimizes the loss on the inner toppings 2) fix the angle as found in step 1 and find the cut radius between tangent to

inner toppings and 6 which minimizes the loss on the outer topping. When searching each of these 1D spaces, we simply take 20 evenly spaced samples and compute the loss (using one of Akshay's utility functions) for each one. Unlike some other groups which search over 3 dimensions at once, our algorithm runs very quickly.

3.3 $n = 4$: An Extension of Our $n = 3$ Solution

Our final goal in the intermediate stage was to extend our $n = 3$ solution to one for $n = 4$. The simplest way to add an extra topping to the current layout while preserving $U = 0$ was to create a 180-degree arc above the inner toppings, where the left half contains one topping and the right half contains another. The radius of this arc was again decided arbitrarily. The topping placements were also all fixed in this stage, so all of our pizzas looked like 6.

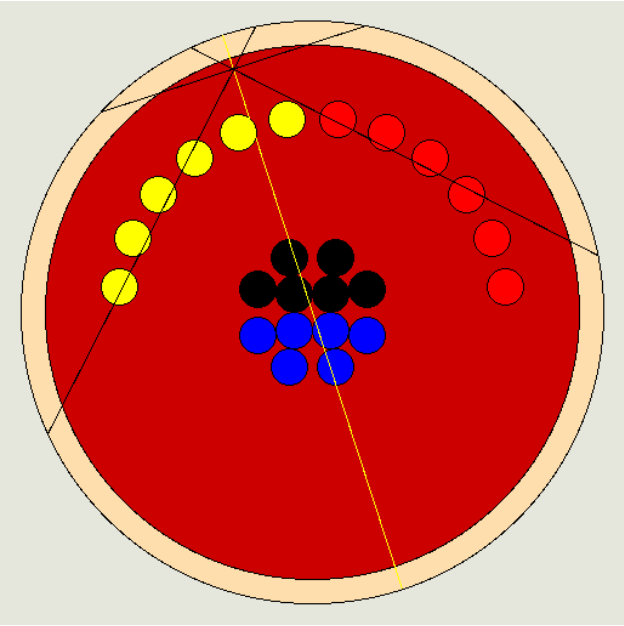

Figure 6: Topping layout and example cut for $n = 4$.

We chose our cut in the same way as $n = 3$ except that in step 2 we minimize the loss on the outer two toppings, plural.

4 Final Design

4.1 Taking Advantage of Symmetry

We discovered in our final iteration that there was one more degree of freedom we could take advantage of. Note that for any cut, rotating it 45 degrees (with origin at cut center) will exactly swap the topping amounts apportioned to customers 1 and 2. Also note that, for our topping layout and cut restrictions, reflecting a cut over the y-axis (with origin at pizza center) exactly swaps the *inner* topping amounts apportioned to customers 1 and 2. Thus, in this same context, reflecting a cut over the y-axis and then rotating it 45 degrees results in the same exact loss on the inner toppings. Moreover, doing so may offer many more options for how the outer toppings get apportioned. This is especially helpful in the $n = 4$ case, since each outer topping is on one side of the pizza. Knowing this, we implemented our cut selection algorithm such that in step 2 we always try both the original minimizing angle found in step 1 as well as 540 degrees minus that angle. We then simply pick which of the two cuts minimizes the loss on the outer toppings. Figure 7 shows an example in which the reflected/rotated cut was chosen. Notice how the yellow source line is rotated 45 degrees from intersecting the pizza center.

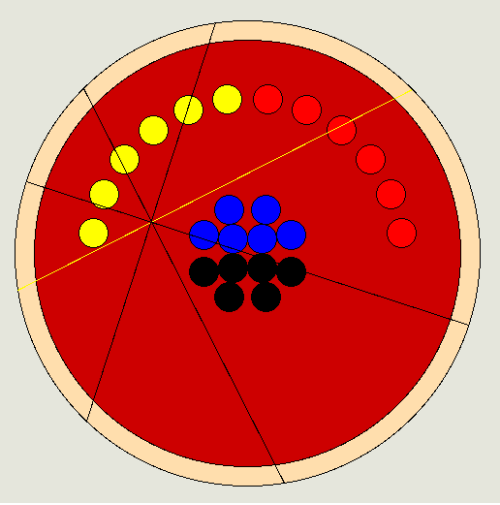

Figure 7: Example cut in which the source line is rotated 45 degrees from intersecting the pizza center.

4.2 Permuting Pizza Toppings Based on Preference Distribution

In all topping cases, our pizza designs are most successful when there are two toppings that are complimentary between customers A and B. If these two complimentary toppings are the center toppings, then our design will see the greatest success. This is because by using only the one diagonal cut to split these two inner toppings we give the same proportion of topping 1 to customer A as we give of topping 2 to customer B. For example, if customer A is given $[.6, .4, ...]$ amount of topping 1 and 2, then customer B is given $[.4, .6, ...]$ of topping 1 and 2. They are directly inverse. Therefore, if the actual preferences are such that they trend in the same way, we will see a greater loss in satisfaction for at least one of the customers.

That being said, it is important to have pizzas with different combinations of center toppings so that we can pick the best pizza to match a specific new customer pair. An initial solution to this was to create permutations of the toppings and evenly choose amongst the toppings. For example, with $n = 3$ the potential pairings for the inside toppings are $(1, 2)$, $(1, 3)$, and $(2, 3)$. So, we initially made 3 pizzas of two pairings and 4 pizzas of the last pairing to sum to 10. Note, order does not matter when selecting the inside toppings (i.e $(1, 2)$ is the same as $(2, 1)$).

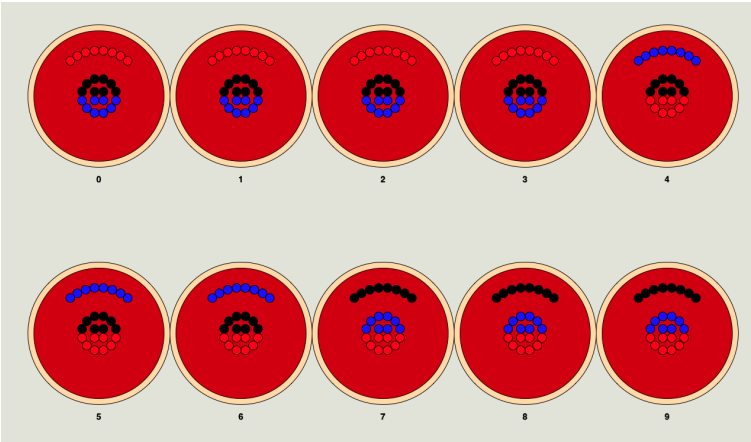

Figure 8: n=3 pizzas with evenly permuted toppings

This was an adequate first approach, as it assumed that the pairings of complementary toppings would be evenly distributed. In the case of a uniformly random distribution tested

in class, this was the case. However, if we want our pizza business to be successful we cannot rely on assumptions. Customers will be happier the more their preferences are satisfied. We can imagine a very real scenario where customers realize that if they can bring a friend who has complimentary preferences, they will gain the most satisfaction. Therefore, we can hope that overtime our customer base will develop to contain complimentary preferences that are not necessarily uniformly distributed. In fact, in reality there are toppings that are naturally complementary like vegetables and meat in the case of a vegetarian and a meat eater. For this reason, we looked into choosing our inner toppings for the 10 pizzas based on the distribution of preferences received at the start of the pizza making.

In order to evaluate a pair of preferences to determine the ideal inner two toppings, we used the following equation where T is the set of topping preferences and a and b represent the two customers:

$$f = |(x_a - y_a) + (y_b - x_b)|, x, y \in T, x \neq y$$

The first term $(x_a - y_a)$ is the difference between a's preference for topping x and for b's. This will be greatest when a likes x more than b. As for the second term, $(y_a - y_b)$ will be negative if b likes y more than a. Therefore, the two ingredients with the greatest value of f will be the most complimentary.

Using this function, for each preference pair in the distribution we recorded what were the two most complimentary toppings. After analyzing 100 preference pairs where the number of toppings is 3, we may end up with the following map of topping pairs to occurences in the distribution: $[[(1,2):50,(1,3):20,(2,3):30]$. Based on this map, 5 pizzas would have toppings $(1,2)$ in the center, 2 pizzas would have $(1,3)$ in the center, and finally 3 pizzas would have $(2,3)$ in the center.

After implementing this change, we tested to see how it changes overall satisfaction across different distributions. The results are recorded in the following table:

	Default	Distribution Based
Random		
n=3	27.44	27.12
n=4	28.77	29.20
Random Cascading		
n=3	44.70	45.96
n=4	52.14	51.856
Alternating Cascading		
n=3	88.08	100.17
n=4	79.92	103.62

Table 1: Average Satisfaction Score By Distribution

The above table records the average satisfaction score taken over 100 games per distribution type for each number of toppings. The case of $n = 2$ is excluded because there is only one way to make a pizza with two toppings in the center. The distributions are defined as follows:

- Random: This is the default distribution tested in class.

- Random Cascading: This distribution is a cascaded uniform distribution where within a given set of preferences, one topping will be heavily biased. It is random because the topping that receives the bias is randomly chosen.

- Alternating Cascading: This distribution fixes the ordering of bias for customer A in ascending order of topping number and for customer B to be descending order of topping number (ex. A: (1,2,3), B: (3,2,1)). The preferences are generated in the same way as the cascading distribution discussed in class, but always in the same order for A and same reverse order for B.

The results from Random and Random Cascading make sense because the toppings that are most complimentary are still random. Hence, there is no statistically significant difference in selecting the two inner toppings based on the distribution versus evenly selecting across permutations (default). It is interesting to note that the average scores in general increase in the Random Cascading distribution, which simply confirms that our pizza design works best

when customer preferences have greater spread and therefore the opportunity to be more complimentary.

The most important result from this table is from the alternating cascading distribution. As described above, this distribution fixes the ingredients such that customer A always likes topping 1 the most and topping 3/4 the least, while customer B always likes topping 3/4 the most and topping 1 the least. In this setting, our algorithm creates all pizzas with the inner two toppings as the complimentary toppings- (1,3) in the n=3 case and (1,4) in the n=4 case. In doing this, our average satisfaction score increases by 30% in the n=4 case and 15% in the n=3 case. This shows that if the distribution of preferences has an uneven distribution of complimentary preferences, our algorithm can capitalize on this difference to greater improve our overall satisfaction scores.

5 Tournament Results

While deciding our pizza making and cutting strategy, we focused on maximizing satisfaction while minimizing uniformity. We also made efforts to do so in a time efficient manner, as pizza is best served hot. With this in mind, our tournament analysis revolves around these three main metrics.

To begin, let us look at the average satisfaction score per distribution for each team. We are Team 5, represented by the green bar in the following bar graphs. D1 refers to the default distribution, D2 refers to Team 1's distribution, D3 refers to Team 3's distribution, and D4 refers to the unseen distribution.

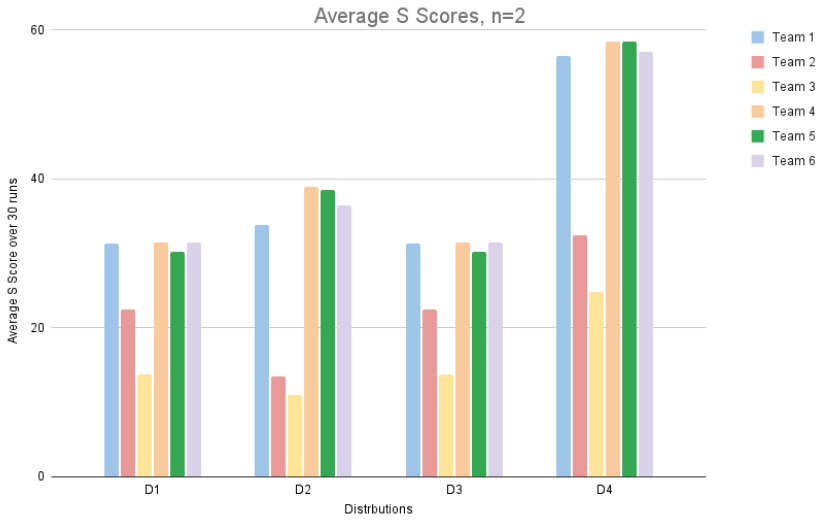

Figure 9: N=2 toppings, Average S Score

With two toppings, most teams implemented our team's original design for $n = 2$ which is why we see comparable scores across the distributions and teams. The very small difference in scores between us and the other top teams may be due to a rounding error, but overall the difference is not statistically significant.

In the case of 3 and 4 toppings, the results are similar. We score consistently high but not the highest, though it is important to note that we do not record a single negative score across all runs in the tournament. For the top four teams, the clustering in scores may be

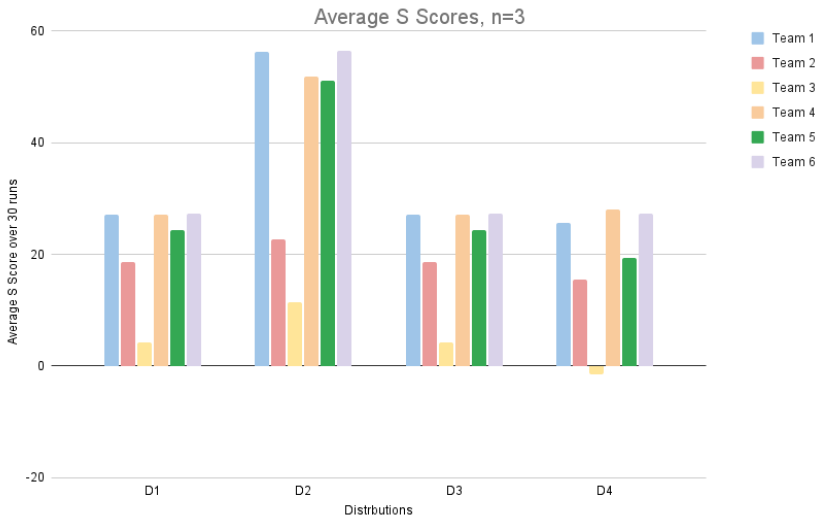

Figure 10: N=3 toppings, Average S Score

Figure 11: N=4 toppings, Average S Score

due to the common adoption of elements of our team's topping placement strategy discussed in class, though other teams built on it in their own way. Either that, or there may be some upper bound on max satisfaction given a set of preferences that the teams are running into. That being said, by analyzing uniformity and time per cut, we can hypothesize why some teams score marginally better in their overall satisfaction scores. The next series of graphs show the average satisfaction minus uniformity scores over the runs:

The story of $n = 2$ remains more or less the same, as uniformity was not too hard to

Figure 12: N=2 toppings, Average S-U Score

maintain for most teams in the $n = 2$ case even without it being their main focus. However, in the $n = 3$ and $n = 4$ case, we start to see more favorable results.

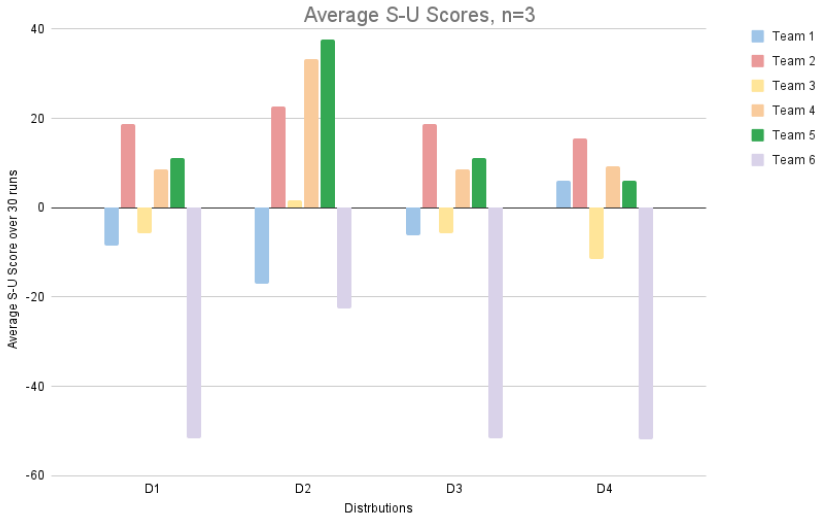

Figure 13: N=3 toppings, Average S-U Score

By taking into account uniformity, our average ranking amongst teams in the $n = 3$ case goes from roughly 4 to 2 and in the $n = 4$ case from roughly 4 to 1.75. This means that compared to the other teams, we are in the top two for effectively balancing satisfaction and uniformity. The last metric we were cognisant of during development is time it takes

Figure 14: N=4 toppings, Average S-U Score

to cut the pizzas. This is where we save the best results for last. The below graphs show average satisfaction minus uniformity divided by the square root of the time it takes to make a cut. Square root was chosen as a way to scale down the magnitude of the run time while preserving the differences between teams' run times.

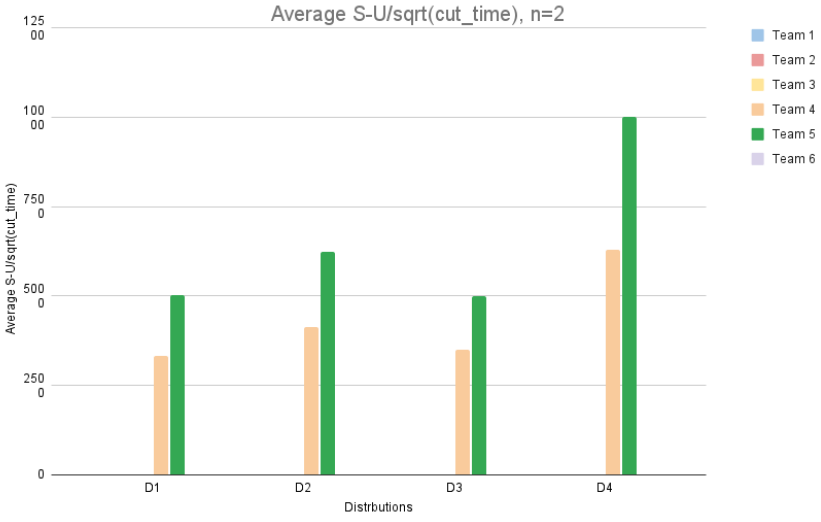

Figure 15: N=2, Average S-U/$\sqrt{cut\_time}$

For $n = 2$, only our team and team 3 had a run time less than 1 second. As a result, dividing by the square root of a number between 0 and 1 caused our scores to be so large

that the other teams do not make it onto the scale. This is an over exaggeration of the difference in run time when comparing average satisfaction minus utility, but still important in demonstrating the speed with which we determine out cuts.

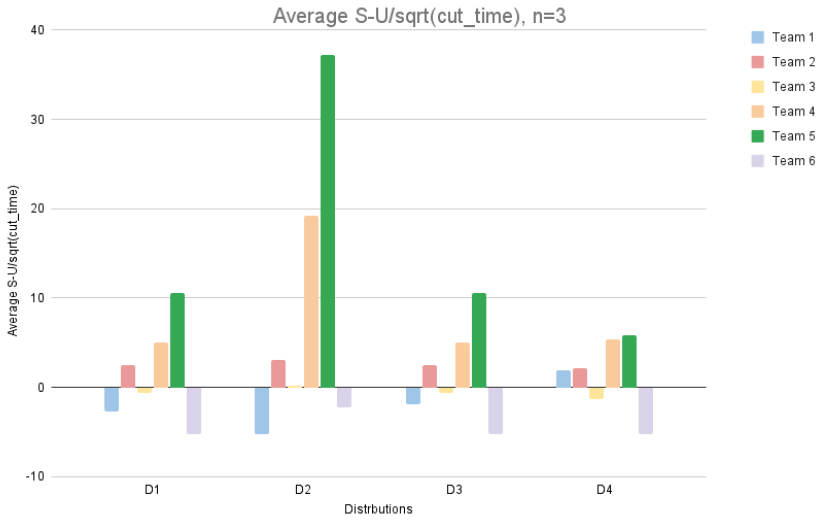

Figure 16: N=3, Average S-U/$\sqrt{cut\_time}$

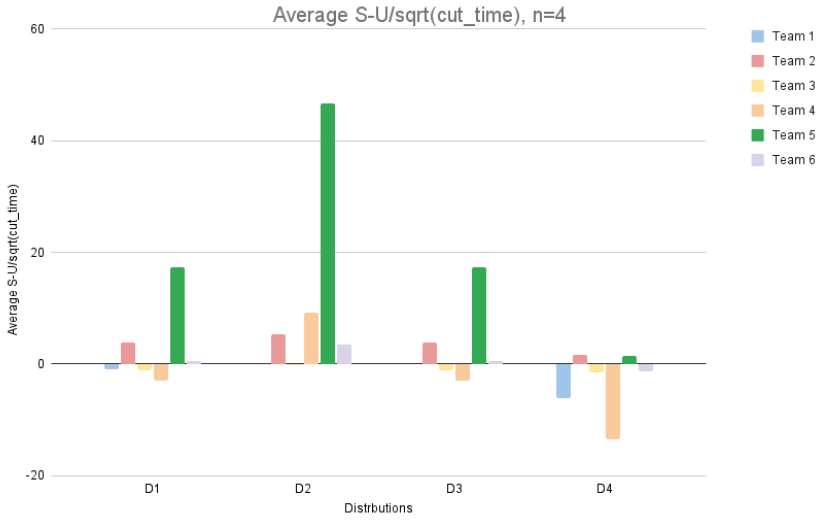

Figure 17: N=4, Average S-U/$\sqrt{cut\_time}$

In the above graphs, our team now ranks number one. When compared to the marginal differences in satisfaction seen in the first set of graphs, our team massively outperforms other teams when factoring in the time to make a pizza cut. In the average satisfaction graphs,

teams had higher scores most likely in part due to large scale brute force search of their pizza cut selection. For our $n = 3$ and $n = 4$ pizza design, we do not claim mathematical optimally like in the $n = 2$ case. Even with our own pizza, there may be a better slice than the slice we cut for a given set of preferences. However, if we take the other teams' scores as evidence, we see that the time spent searching for this perfect cut given the pizza and preferences is potentially not worth the small gain we see in satisfaction.

6 Discussion

6.1 Conclusions

In summary, we serve our pizza with competitive satisfaction and uniformity scores, and we do so significantly quicker than the competition. Our commitment to mathematical precision and carefully crafted pizza designs allowed us to achieve high satisfaction scores and low uniformity while keeping run time per cut trivial. Our tournament results highlight our success in these metrics we chose to measure against. While our preference distribution approach to selecting inner toppings did not seem to play a measurable role in the tournament's distributions, it shows our commitment to meet our customers' needs and use all available information at hand. Ultimately, in a city where time is money, our pizza strategy would be hard to beat.

6.2 Future Work

Like any good open ended project, there are many opportunities for future work. We focused on S and U for this project. Going forward, we could see how to adapt our design to improve the slice metric or center offset. We could also look at how to improve S if we loosen our commitment to maintaining a low U. Another area for future work would be utilizing the distribution information more. Our current inner topping selection is a naive approach that does not consider standard deviation or other statistical factors that may make our results more robust. Additionally, we could explore how different distributions may affect our pizza designs at a more fundamental level. For example, in the $n = 4$ scenario it is important that the two middle toppings are complementary. If the distribution shows that customer's preferences are pretty evenly matched, perhaps we could go with a different cut or topping layout strategy.

7 Summary of Contributions

- Emma: Implemented $n = 3$ based on team's agreed design and permuting pizza toppings based on preference distribution. Wrote sections in paper on permuting pizza toppings based on dist., tournament analysis, and discussion.
- Fenglei: Designed $n = 2, 3, 4$ solutions, implemented $n = 2$ solution and starter codes for $n = 3, 4$ solutions. Wrote intro, initial design, and acknowledgement parts of the report.
- Justin: Implemented intermediate and final solutions for $n = 4$. Wrote sections in paper and report on intermediate solutions and symmetry.

8 Acknowledgements

We would like to thank Professor Kenneth Ross and the teaching assistant for bringing this smart designed problem. We have intellectually benefited a lot from brainstorming for solving this problem. We would also like to extend our thanks to our classmates, who generously shared their ideas in class discussions which definitely helped shaping our design and fine-tuning our algorithms.