

# CS 251 - Betweenness (26.14)

Sarah Eisenstat (seisenst)

December 5, 2008

## 1 Random Ordering

For each of the triplets  $t$  in  $T$ , define a random variable  $X_t$  that is equal to 1 if the randomly selected ordering satisfies  $t$ , and 0 otherwise. Then define a random variable  $Y$  that is equal to the total number of satisfied triplets in the random ordering. This means that we have:

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{t \in T} X_t\right] = \sum_{t \in T} \mathbb{E}[t]$$

What is  $\mathbb{E}[t]$ ? Well, these are all of the possible orderings that we could have of a triplet  $t = (a, b, c)$ :  $(a, b, c)$ ,  $(a, c, b)$ ,  $(b, a, c)$ ,  $(b, c, a)$ ,  $(c, a, b)$ ,  $(c, b, a)$ . Two of those six possible orderings satisfy  $t$ . Because we have chosen a random permutation of the elements, we know that any subset of those elements is also a random permutation. In other words, the permutation of  $a, b, c$  is chosen uniformly at random. This means that with probability  $2/6 = 1/3$ , the permutation of  $a, b, c$  will satisfy  $t$ , and otherwise it will not. Therefore,  $\mathbb{E}[t] = 1/3$ . Hence:

$$\mathbb{E}[Y] = \sum_{t \in T} \mathbb{E}[t] = \sum_{t \in T} \frac{1}{3} = \frac{|T|}{3}$$

In other words, the expected number of triplets is exactly one third the total number of triplets.

## 2 Derandomization and Upper Bound

In order to derandomize the algorithm, we cannot simply take the conditional expectation for each possible permutation, because that would take factorial time. Instead, we rewrite the algorithm as follows:

```
set  $i = 1$ 
while  $|S| \geq 0$ 
  select  $s$  uniformly at random from  $S$ 
  set  $p(i) = s$ 
  set  $S = S \setminus \{s\}$ 
  set  $i = i + 1$ 
output  $p$ 
```

Then we can derandomize by selecting the  $s \in S$  that maximizes the expected value of the output, given the choices that we have made so far. There are clearly a polynomial number of conditional expected values to calculate at each step (one for each of the remaining elements of  $S$ ), but can we calculate them in polynomial time? Well, consider some triple  $t \in T$ . We know that zero, one, two, or three of the elements of that triple have had their position selected already. Let's break this into cases:

- **Case 0:** Because we assume that the rest of the permutation will be selected randomly, we know that each permutation on  $t$  will occur with equal probability. Therefore:

$$E[X_t \mid p(1), \dots, p(k-1)] = 1/3$$

- **Case 1:** If the only element selected so far is the middle element of the triple, then we know that this triple cannot be satisfied no matter how the other two are arranged, and so  $E[X_t \mid p(1), \dots, p(k-1)] = 0$ . Otherwise, we know that there are two possible ways to order the remaining elements, and exactly one of those will satisfy  $t$ . Therefore, in this case we have  $E[X_t \mid p(1), \dots, p(k)] = 1/2$ .
- **Case 2:** In this case, we already know the ordering of the three elements in  $t$ , even if we don't know the exact positioning. Therefore, we already know whether the triple is satisfied, and so we can easily calculate whether the expected value is 0 or 1.
- **Case 3:** By similar argument to case 2, it's easy to calculate whether we have already satisfied  $t$  with our permutation.

Therefore, we can calculate the conditional expected value of each individual triple  $t$ . This means that the total conditional expected value is also easy to calculate, and so we know that we can perform the derandomization. We know that at every step, we increase the conditional expected value, and so we have:

$$COST(OUTPUT) \geq E[Y] = \frac{|T|}{3} \geq \frac{OPT}{3}$$

This gives us an approximation factor of  $1/3$ .

Consider the following betweenness problem:

$$S = \{a, b, c\}$$

$$T = \{(a, b, c), (c, a, b), (b, c, a)\}$$

For each of those triples, there are exactly two permutations of  $a, b, c$  that satisfy it. Those permutations are not overlapping. Therefore, any given permutation of  $S$  can only satisfy one triple. So no matter what the algorithm is, it cannot return an ordering that satisfies more than one triple. Therefore, no matter what the algorithm is, it cannot return a permutation that satisfies more than one third of the triples given. Therefore, this is an upper bound on any approximation algorithm, if we use the number of triples as the target.

### 3 Quadratic Program

Say that we have a satisfiable instance of the betweenness problem. This means that we have a ranking that satisfies all of the triples. Now we wish to show that we can construct a solution to the quadratic program. To do so, we let  $p_i$  be the index of  $x_i$  in the ordering. Does this give us a feasible solution to the quadratic program? Well, because the values of the  $p_i$ 's vary over the integers, and no value is shared by different items, we know that for any  $i \neq j$ ,  $p_i$  and  $p_j$  differ by at least one, and so the square of the difference is  $\geq 1$ . This means that we have satisfied the constraint  $(p_i - p_j)^2 \geq 1$ . What about the second constraint? Well, consider some triple  $(x_i, x_j, x_k) \in T$ . Because we have a satisfiable ordering, we know that one of two cases must hold:

- **Case 1:**  $x_i < x_j < x_k$ . Then  $p_i < p_j$  and  $p_j < p_k$ , by the way we have selected the values of  $p$ . With a little algebraic manipulation, we get  $p_i - p_j < 0$  and  $p_k - p_j > 0$ . If we then take the product of  $(p_i - p_j)$  and  $(p_k - p_j)$ , we know that we're multiplying a positive number by a negative number, which will result in a negative number. Therefore,  $(p_i - p_j)(p_k - p_j) \leq 0$ , and so the constraint is satisfied.
- **Case 2:**  $x_i > x_j > x_k$ . Then  $p_i - p_j > 0$  and  $p_k - p_j < 0$ . Once again, the product of a positive number and a negative number is negative, and so  $(p_i - p_j)(p_k - p_j) \leq 0$ , and so the constraint is satisfied.

This means that any solution to the betweenness problem will give us a solution to the quadratic program. What about the reverse?

Say that we have a solution for the quadratic program. Now we wish to construct an ordering that satisfies all the triples of the betweenness problem. To do so, we sort the values of  $S$  using  $p_i$  to rank them. Does this satisfy all of the triples? Well, consider some triple  $t = (x_i, x_j, x_k)$ . Because we have a solution to the quadratic program, we know that  $(p_i - p_j)(p_k - p_j) \leq 0$ . By our first constraint, we know that  $p_i - p_j \neq 0$  and  $p_k - p_j \neq 0$ . This means that one of those must be positive, and one of those must be negative. This gives us two possibilities:

- **Case 1:**  $p_i - p_j < 0$  and  $p_k - p_j > 0$ . In other words,  $p_i < p_j < p_k$ . Because we then order the corresponding  $x_i, x_j, x_k$  as  $x_i < x_j < x_k$ , we know that  $t$  is satisfied.
- **Case 2:**  $p_i - p_j > 0$  and  $p_j - p_k < 0$ . In other words,  $p_i > p_j > p_k$ . This means that, because the  $x$ 's are ordered in the same way as the  $p$ 's, we have satisfied  $t$ .

Therefore, this gives us an ordering that satisfies every triple, and so we have shown that the betweenness problem was satisfiable.

Putting these two things together, we get that the two problems are equivalent.

### 4 Vector Program and Semidefinite Program

The vector programming relaxation has the constraints:

$$\begin{aligned} (v_i - v_j)^2 &\geq 1 \text{ for all } i \neq j \\ (v_i - v_j)(v_k - v_j) &\geq 0 \text{ for all } (x_i, x_j, x_k) \in T \end{aligned}$$

If we expand those multiplications, we get:

$$\begin{aligned} v_i \cdot v_i - 2v_i \cdot v_j + v_j \cdot v_j &\geq 1 \text{ for all } i \neq j \\ v_i \cdot v_k - v_i \cdot v_j - v_j \cdot v_k + v_j \cdot v_j &\geq 0 \text{ for all } (x_i, x_j, x_k) \in T \end{aligned}$$

Then the semidefinite program becomes:

$$\begin{aligned} y_{ii} - 2y_{ij} + y_{jj} &\geq 1 \text{ for all } i \neq j \\ y_{ik} + y_{jj} - y_{ij} - y_{jk} &\geq 0 \text{ for all } (x_i, x_j, x_k) \in T \end{aligned}$$

## 5 Unsatisfiable Example

Consider the following betweenness problem:

$$\begin{aligned} S &= \{a, b, c, d\} \\ T &= \{(a, b, c), (b, c, d), (d, a, b)\} \end{aligned}$$

It is clearly not satisfiable, because those three triples mean that neither  $a$  nor  $b$  nor  $c$  can occur as the highest or lowest element, leaving only  $d$  to serve as both highest and lowest. This is, of course, a contradiction. However, the corresponding vector program is:

$$\begin{aligned} (v_a - v_b)^2 &\geq 1 \\ (v_a - v_c)^2 &\geq 1 \\ (v_a - v_d)^2 &\geq 1 \\ (v_b - v_c)^2 &\geq 1 \\ (v_b - v_d)^2 &\geq 1 \\ (v_c - v_d)^2 &\geq 1 \\ (v_a - v_b)(v_c - v_b) &\leq 0 \\ (v_b - v_c)(v_d - v_c) &\leq 0 \\ (v_d - v_a)(v_b - v_a) &\leq 0 \end{aligned}$$

I claim that the following is a feasible solution to the vector program:

$$\begin{aligned} v_a &= (1, 0, 0, 1) \\ v_b &= (1, 1, 0, 0) \\ v_c &= (0, 1, 1, 0) \\ v_d &= (0, 0, 1, 1) \end{aligned}$$

We can check each of the constraints:

$$\begin{aligned}
(v_a - v_b)^2 &= (0, -1, 0, 1)^2 = 2 \geq 1 \\
(v_a - v_c)^2 &= (1, -1, -1, 1)^2 = 4 \geq 1 \\
(v_a - v_d)^2 &= (1, 0, -1, 0)^2 = 2 \geq 1 \\
(v_b - v_c)^2 &= (1, 0, -1, 0)^2 = 2 \geq 1 \\
(v_b - v_d)^2 &= (1, 1, -1, -1)^2 = 4 \geq 1 \\
(v_c - v_d)^2 &= (0, 1, 0, -1)^2 = 2 \geq 1 \\
(v_a - v_b)(v_c - v_b) &= (0, -1, 0, 1) \cdot (-1, 0, 1, 0) = 0 \leq 0 \\
(v_b - v_c)(v_d - v_c) &= (1, 0, -1, 0) \cdot (0, -1, 0, 1) = 0 \leq 0 \\
(v_d - v_a)(v_b - v_a) &= (-1, 0, 1, 0) \cdot (0, 1, 0, -1) = 0 \leq 0
\end{aligned}$$

From this solution to the vector problem, we can construct the solution to the semidefinite program by setting  $Y_{ij} = v_i^T v_j$ , or the matrix:

$$\begin{array}{cccc}
2 & 1 & 0 & 1 \\
1 & 2 & 1 & 0 \\
0 & 1 & 2 & 1 \\
1 & 0 & 1 & 2
\end{array}$$

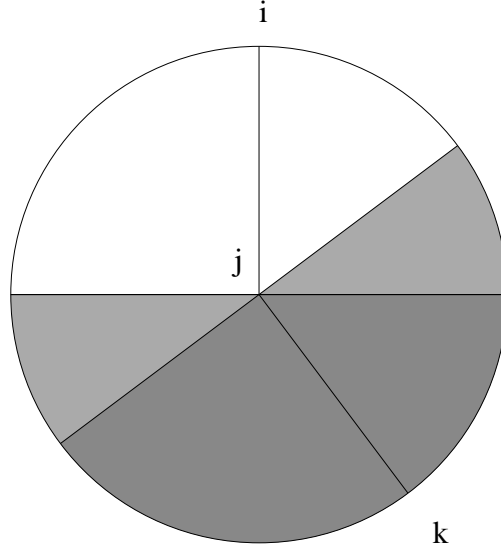
## 6 Analysis of Randomized Rounding

In order to determine the expected number of satisfied triplets, we must determine the probability that some triplet  $t = (x_i, x_j, x_k) \in T$  is satisfied by the ordering that we output. It will only be satisfied if  $r \cdot v_i < r \cdot v_j < r \cdot v_k$  or  $r \cdot v_i > r \cdot v_j > r \cdot v_k$ .

Because of the second set of constraints for the vector program, we know that  $(v_i - v_j)(v_k - v_j) \leq 0$ . Because the dot product of two vectors is equal to the product of their sizes and the cosine of the angle between them, we know that the cosine must be negative (because the magnitudes cannot be). That means that the angle between  $v_i - v_j$  and  $v_k - v_j$  must be at least  $\pi/2$ .

$v_i$ ,  $v_j$ , and  $v_k$  form a triangle on the plane that they define. Project  $r$  onto that plane. Because  $r$  was selected uniformly at random from the unit sphere, the direction of  $r$  in the plane will also be selected uniformly at random. For what directions will  $r$  yield an ordering that satisfies the triple?

Say that the final ordering has  $r \cdot v_i < r \cdot v_j < r \cdot v_k$ . This means that  $r \cdot (v_i - v_j) < 0$  and  $r \cdot (v_k - v_j) > 0$ . For what angles of  $r$  does that hold true? Well, each of those statements represents a semicircle of possible values for  $r$  to take on. Their overlap is the set of values where both statements hold. How large is the overlap?



The dark gray area on that diagram is the overlap. Using some elemental geometry, we find that the angle of that area is equal to the size of the angle between  $v_i - v_j$  and  $v_k - v_j$ . (This is because the constraint  $r \cdot (v_k - v_j) > 0$  is a semicircle centered around  $v_k - v_j$ , while the constraint  $r \cdot (v_i - v_j) < 0$  cuts out any portion of that semicircle closer than  $\pi/2$  to  $v_i - v_j$ . The light gray slice on the left, therefore, is equal to  $\pi/2 - (\theta_{ik} - \pi/2) = \pi - \theta_{ik}$ , which means that when we remove it from our size  $\pi$  semicircle, we are left with  $\theta_{ik}$ .)

Say that the final ordering has  $r \cdot v_i > r \cdot v_j > r \cdot v_k$ . This means that  $r \cdot (v_i - v_j) > 0$  and  $r \cdot (v_k - v_j) < 0$ . By analogous argument to the above, we may say that the size of the angles that fit those requirements is equal to the angle between  $v_i - v_j$  and  $v_k - v_j$ . Since we know that that angle must be  $\geq \pi/2$ , we know that the set of “good”  $r$ s can have twice that size, and so the probability of satisfying a single  $t$  is at least  $\pi/(2\pi) = 1/2$ . Therefore, the expected number of satisfied triplets is  $|T|/2$ .