7-2021

Geometric Analysis Leads to Adversarial Teaching of Cybersecurity

Christian Servin
El Paso Community College, cservin1@epcc.edu

Olga Kosheleva
University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-20-78c

Geometric Analysis Leads to Adversarial Teaching of Cybersecurity

Christian Servin, Olga Kosheleva, and Vladik Kreinovich

Abstract As time goes, our civilization becomes more and more dependent on computers and therefore, more and more vulnerable to cyberattacks. Because of this threat, it is very important to make sure that computer science students – tomorrow's computer professionals – are sufficiently skilled in cybersecurity. In this paper, we analyze the need for teaching cybersecurity from the geometric viewpoint. We show that the corresponding geometric analysis leads to adversarial teaching – an empirically effective but not-well-theoretically-understood approach, when the class is divided into sparring mini-teams that try their best to attack each other and defend from each other. Thus, our analysis provides a theoretical explanation for the empirical efficiency of this approach.

1 Need for Teaching Cybersecurity: The Practical Problem and Its Natural Geometric Formalization

Cybersecurity is important. As time goes, our world relies on computers more and more. Computers are needed for us to communicate, to pay and get paid, to travel – in short, for all aspects of our lives. The ubiquity of computers makes all these aspects vulnerable to cyberattacks. This, in turn, makes cybersecurity – the ability to defeat such attacks – extremely important. It is therefore very important to teach cybersecurity skills to computer science students – and to other future (and present) computer professionals.

Christian Servin
Computer Science and Information Technology Systems Department
El Paso Community College (EPCC), 919 Hunter Dr., El Paso, TX 79915-1908, USA
cservin1@epcc.edu

Olga Kosheleva and Vladik Kreinovich
University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA
e-mail: olgak@utep.edu, vladik@utep.edu

A natural geometric approach to cybersecurity. During the cybersecurity training, students are exposed to several possible situations. Let us denote the number of such situations by n, and the situations themselves by $s_1, \ldots, s_n$. Let us also denote by S the set of all possible situations.

In real life:

- when the correspondingly trained person encounters a situation s which is similar to one of the situations s_i with which he/she is familiar,
- then, with high probability, the person will be able to adequately react to the new situation s as well.

The closer the situations s and s_i, the higher the probability that this person's reaction will be adequate. Vice versa, the larger the difference between the two situation, the higher the probability that the cyberattack will be successful.

It is usually possible to describe the degree of difference between the two situations s and s' by a number $d(s, s')$. This number can be determined by experts, or it can be based on the empirical data. In this paper, we will assume that these numbers are determined in such a way that the following natural properties are satisfied:

- $d(s, s) = 0$ for all s,
- $d(s, s') > 0$ for all $s \neq s'$,
- $d(s, s') = d(s', s)$ for all s and s', and
- the triangle inequality $d(s, s'') \leq d(s, s') + d(s', s'')$ for all s, s', and s''.

In mathematical terms, this means that the corresponding function $d(s, s')$ is a *metric*.

The closer a new situation s to one of the previous situations s_i, i.e., the smaller the value

$$d(s, \{s_1, \ldots, s_n\}) \stackrel{\text{def}}{=} \min_i d(s, s_i), \tag{1}$$

the more efficient the defence against this situation. Vice versa, the larger the value (1), the less efficient the defense, and the larger the probability that the attack will succeed.

The attackers are usually familiar with what potential defenders know. So, to attack, they create a situation s for which the probability of success is the largest, i.e., equivalently, for which the value (1) is the largest. Thus, the probability that an attack will be successful is determined by the value

$$\max_{s \in S} d(s, \{s_1, \ldots, s_n\}). \tag{2}$$

The larger this value, the larger the probability that the attack will be successful.

In mathematical terms, the maximum (2) can be naturally described in terms of the *Hausdorff distance* between the two sets A and B. This distance is defined as the smallest value $\varepsilon > 0$ for which:

- for each element $a \in A$, there exists an element $b \in B$ for which $d(a, b) \leq \varepsilon$, and
- for each element $b \in B$, there exists an element $a \in A$ for which $d(a, b) \leq \varepsilon$.

In this terms, the value (2) is equal to the *Hausdorff distance*

$$d_H(S, \{s_1, \ldots, s_n\}) \tag{3}$$

between:

- the set S of all possible situations, and
- the set $\{s_1, \ldots, s_n\}$ on which the defenders have been trained.

The larger the number n of situations that the student study, the better the students will be prepared to defend against cyberattacks. However, in practice, the number n of different situations is limited by the need to describe all of them within the given training time. From this viewpoint, it is reasonable to assume that the value n is fixed. In this case, our objective is:

- *given* n,
- to *minimize* the probability of the attack's success, i.e., equivalently, to find the situations $s_1, \ldots, s_n$ for which the expression (2)-(3) is the smallest.

We will denote the optimal value of the expression (2)-(3) by $\varepsilon_n(S)$:

$$\varepsilon_n(S) \stackrel{\text{def}}{=} \min_{s_1,\ldots,s_n} d_H(S, \{s_1, \ldots, s_n\}). \tag{4}$$

In this paper, we analyze how this minimization problem can be solved.

An alternative formulation. Alternatively, we can consider a different version of this problem: instead of fixing the training time (and thus, fixing n):

- we can fix the desired probability that the defense will be successful – i.e., equivalently, the desired value $\varepsilon > 0$ of the quantity (2) – and
- we want to find out how to train defenders to reach this efficiency within the smallest possible time – i.e., via the smallest possible number of training situations n.

In this alternative formulation, we want to find the set of situations $\{s_1, \ldots, s_n\}$ for which every other situation $s \in S$ is ε-close to one of them, i.e., for which, for each $s \in S$, there exists an i for which $d(s, s_i) \le \varepsilon$.

In mathematical terms, such a set is known as an *ε-net*; see, e.g., [7, 8]. The smallest possible number of elements in an ε-net for a set S is usually denoted by $N_\varepsilon(S)$. The binary logarithm $H_\varepsilon(S) \stackrel{\text{def}}{=} \log_2(N_\varepsilon(S))$ of this number is called *ε-entropy*; see, e.g., [7, 8].

The two formulations are equivalent. From the mathematical viewpoint, both formulation represent, in effect, the same optimization problem. Indeed, one can check that $H_\varepsilon(S)$ is an inverse function to $\varepsilon_n(S)$, in the following precise sense:

$$H_\varepsilon(S) \le n \Leftrightarrow \varepsilon \ge \varepsilon_n(S).$$

We cannot feasibly solve the corresponding optimization problem. It is known that problem of finding the smallest ε-net is, in general, NP-hard; see, e.g., [1].

This means, in particular, that unless P = NP (which most computer scientists believe to be false; see, e.g., [3]), it is not possible to have a feasible algorithm that:

- given a set S with a metric $d(s, s')$,
- always returns the optimal ε-net for this set.

So, we cannot rely on any general feasible algorithm to find the optimal set of training situations.

The best we can do is to find a solution which is as *close to optimal* as feasibly possible.

Comment. This paper is a revised and extended version of our shorter conference paper [10]; specifically, we made the following two extensions:

- first, in this paper, we explain the geometric model in more detail that in the conference version;
- second, we also explain how to best implement the corresponding teaching strategy.

2 A Natural Algorithm for Solving Our Problem

A natural algorithm. In the alternative formulation, we do not know how many training situations we will need to achieve the desired proficiency, i.e., the given value ε. So, let us add such situations one by one, until we reach the desired proficiency.

We start with selecting the first situation s_1. If this the only situation we learn, then the quality of this learning can be described by the value $d_H(S, \{s_1\}) = \max\limits_{s} d(s, s_1)$. So, we need to find the situation $s_1 \in S$ for which this value is the smallest possible. In mathematics, such a point s_1 is known as a *center* (or a *Chebyshev center*) of the set S.

If for this center, we already have $d_H(S, \{s_1\}) \le \varepsilon$, then we are done. But what if this inequality has not been reached yet? In this case, as we have mentioned, the attacker can reach the highest probability of the attack's success if he/she selects a situation s_2 for which the distance $d(s_1, s_2)$ is the largest:

$$d(s_1, s_2) = \max_{s} d(s_1, s).$$

A natural idea is therefore to train the students on this most-probable attack situation. This way, the most potentially damaging attack will be prevented.

Similarly:

- if we have already trained the students on situations $s_1, \ldots, s_n$, and we still have not yet reached the desired level of proficiency ε, i.e., if $\max\limits_{s} d(s, \{s_1, \ldots, s_n\}) > \varepsilon$,

- then a natural idea is to next train the students on the situation s_{n+1} which is potentially the most damaging, i.e., for which

$$d(s_{n+1}, \{s_1, \ldots, s_n\}) = \max_s d(s, \{s_1, \ldots, s_n\}). \tag{5}$$

We continue this procedure until we reach the desired proficiency level, i.e., until we reach the desired inequality

$$\max_s d(s, \{s_1, \ldots, s_n\}) \le \varepsilon. \tag{6}$$

Comment. Note that we only select the next situation s_{n+1} if

$$\max_s d(s, \{s_1, \ldots, s_n\}) > \varepsilon.$$

In this case, the situation s_{n+1} is selected so as to maximize this value. So, we have $d(s_{n+1}, \{s_1, \ldots, s_n\}) > \varepsilon$, i.e., $d(s_{n+1}, s_i) > \varepsilon$ for all $i \le n$.

In general, this means that if we select the training situations s_i by using the above algorithm, then we will have $d(s_i, s_j) > \varepsilon$ for all $i \ne j$. (So, all training situations generated by our algorithm are reasonably different.)

We stop when $d_H(S, \{s_1, \ldots, s_n\}) \le \varepsilon$. Thus, we have

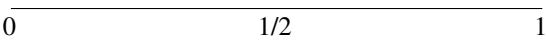
$$d(s_i, s_j) > d_H(S, \{s_1, \ldots, s_n\})$$

for all $i \ne j$.

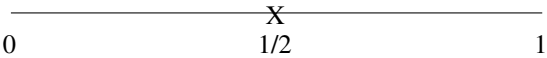
3 Natural Algorithm: Visualization

To make it easier to understand, let us give two simple geometric illustrations of the above algorithm. These examples are similar to examples provided in [6] for a different application of a similar idea – to selecting benchmarks for testing different numerical algorithms.

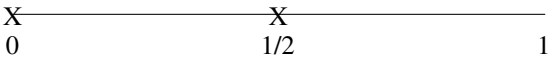
1D example. Let us start with the simplest example of a metric space S – namely, the interval $[0, 1]$:



It is reasonable to select the midpoint 1/2 as s_1:

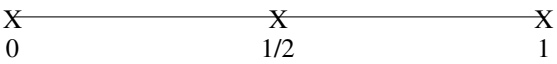


There are two points that are the farthest from s_1: the left endpoint 0 and the right endpoint 1. Without losing generality, let us select $s_2 = 0$:

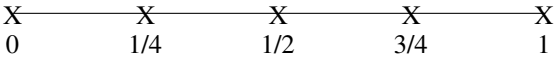


Now, $s_3 = 1$ is the point with the largest value of

$$d(s, \{s_1, s_2\}) = \min(d(s, s_1), d(s, s_2)) :$$



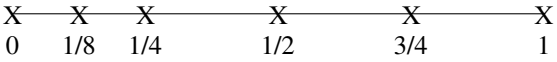
At this stage, the midpoints between 0 and 1/2 and between 1/2 and 1 are the farthest from the set $\{s_1, s_2, s_3\} = \{0, 1/2, 1\}$, so, after two stages, we add them both:



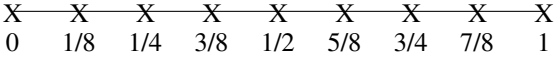
Now, the largest possible value of

$$d(s, \{s_1, s_2, s_3, s_4, s_5\}) = d(s, \{0, 1/4, 1/2, 3/4, 1\})$$

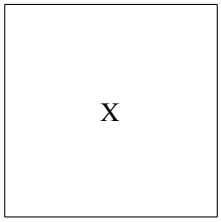
is 1/8. So, at the next stage, we add one of the points in between the existing ones, e.g., the first one (1/8):



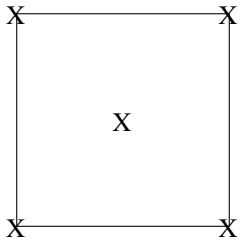
After three more stages, we add all midpoints, so we arrive at the following configuration:



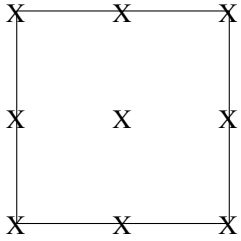
2D example: square. For a unit square, we get a similar situation. First, let us pick the midpoint as s_1:



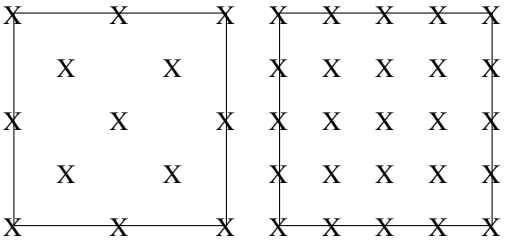
Then, the next four selections s_i are the vertices:



After this, the next four selected points s_i are he midpoints of the four edges:



Here, we have, in effect, four sub-squares. On the next stage, the same procedure is repeated for each sub-square, etc.



4 Our Algorithm Is Efficient

Our algorithm eventually leads to success. Indeed, the set S of possible types of situations is finite. At each step, we add one more select to this set. So, the number of steps cannot exceed the number of elements in the set S – in other words, this algorithm will eventually stop.

Our algorithm is asymptotically optimal. Our algorithm is feasible, so it cannot be optimal – since our problem is NP-hard. Indeed, one can find examples of metric spaces in which this algorithm does not lead to the optimal arrangement.

It is possible to prove, however, that this algorithm is *asymptotically optimal* in the following precise sense. Recall that for each n, our objective is to minimize the value $d_H(S, \{s_1, \ldots, s_n\})$, i.e., make it as close as possible to the smallest possible value $\varepsilon_n(S)$ of this quantity.

- Due to NP-hardness, we cannot feasibly find the optimizing situations s_i for which $d_H(S, \{s_1, \ldots, s_n\})$ is exactly equal to $\varepsilon_n(S)$.
- It turns out that our algorithm increases the value ε, crudely speaking, by no more than a factor of 2.

To be more precise, for the situations s_i generated by our algorithm, the following inequality is satisfied:

$$d_H(S, \{s_1, \ldots, s_n\}) \le 2\varepsilon_{n-1}(S).$$

Let us prove this inequality by contradiction. Indeed, let us assume that, vice versa, $d_H(S, \{s_1, \ldots, s_n\}) > 2\varepsilon_{n-1}(S)$, i.e., that

$$\varepsilon_{n-1}(S) < \frac{1}{2} \cdot d_H(S, \{s_1, \ldots, s_n\}),$$

and let us show that this assumption leads to a contradiction.

By definition of $\varepsilon_{n-1}(S)$, this means that in the set S, there exists a $\varepsilon_{n-1}(S)$-net $s'_1, \ldots, s'_{n-1}$. By the definition of an ε-net, this means, in particular, that for every i, there exists an index i' for which

$$d(s_i, s'_{i'}) < \frac{1}{2} \cdot d_H(S, \{s_1, \ldots, s_n\}).$$

Let us denote one of such indices by $e(i)$; then, we have

$$d(s_i, s'_{e(i)}) < \frac{1}{2} \cdot d_H(S, \{s_1, \ldots, s_n\}).$$

For two different values i and j, we cannot have $e(i) = e(j)$ – otherwise, we will have

$$d(s_i, s_j) \le d\left(s_i, s'_{e(i)}\right) + d\left(s'_{e(i)}, s_j\right) <$$

$$\frac{1}{2} \cdot d_H(S, \{s_1, \ldots, s_n\}) + \frac{1}{2} \cdot d_H(S, \{s_1, \ldots, s_n\}) = d_H(S, \{s_1, \ldots, s_n\}).$$

On the other hand, we have shown, in the comment after formula (6), that we must have

$$d(s_i, s_j) > d_H(S, \{s_1, \ldots, s_n\}).$$

for all $i \ne j$. Thus, for different value $i \ne j$, we must have $e(i) \ne e(j)$.

So:

- to each of n situations s_i ($i = 1, \ldots, n$), there corresponds a different situation $s'_{e(i)}$,
- however, this is impossible, since this would mean that we have at least n different situations s'_j, and we only have $n - 1$ of them.

This contradiction shows that our assumption

$$\varepsilon_{n-1}(S) < \frac{1}{2} \cdot d_H(S, \{s_1, \ldots, s_n\})$$

is false. So, tthe desired inequality is proven.

5 How Can We Implement This Algorithm

How can we implement the above algorithm: idea. Selecting s_1 may be a problem, but once we have already selected $s_1, \ldots, s_n$, selecting the next situation s_{n+1} is straightforward. Indeed, as we have mentioned, this is exactly how the actual attacker naturally selects the way to attack: by selecting the situation s for which his/her probability of success is the largest, i.e., by selecting the situation s_{n+1} as described by the formula (5). Thus, all we have to do to implement this algorithm is to divide students into competing teams, and let the team which is currently attacking select the next situation.

This leads to the following way to implement the above algorithm in teaching cybersecurity.

Resulting teaching strategy. The instructor divides the class into one or more pairs of sparring mini-teams. In each pair, the teams interchangingly try to attack each other and to defend their team from a other team's attack.

This way, no matter what the original situation s_1 is, after each n situations $s_1, \ldots, s_n$, the attacking team will naturally select the situation s_{n+1} that satisfies the property (5).

This strategy has been tried – and it works. This teaching strategy has indeed been tried – under the name of *adversarial teaching* – and it works; see, e.g., [4, 5].

A similar strategy works in engineering design. Interestingly, a similar approach works in military engineering as well. For example, according to [9], new fighter planes are designed as follows – by using a program that simulates dogfights between different planes:

- The first stage is natural: we consider several possible designs, and for each of them, we simulate how this design will perform in a possible confrontation with the fighter planes used by the existing adversaries. We continue doing this until we find a design that can beat all the possible opponents.
- At first glance, this may seem to be sufficient, but, on second thought, it is not: it is not enough for a future plane to be better that what the opponent has now, we need to have a design that will be better than what the opponent will have in the future. To design such a plane, we perform the second stage of the design process: namely, we design a plane that will be better than not only the current planes, but also better than our first-stage design.
- Then, we design a plane that will be better than the second-stage design, etc.

At the end, we get an almost perfect future plane – and this is what is then implemented and tested.

Our geometrical model explains why this strategy works. Adversarial teaching works, but why it works has, so far, remained largely a mystery.

In this paper, we explain why it works. Moreover, we explain that this strategy leads to asymptotically optimal results.

Comment. The above text confirms that a healthy and productive competition between students is good for education.

This is a known fact of education is general, not just in teaching cybersecurity; see, e.g., [2]. Moreover, competition is good in general, not just for learning.

Indeed, competition means that if one team is somewhat ahead, the other team tries to catch up, and vice versa. Without a competition, the folks may stagnate and not progress too much. Theoretically, this can also happen in a competitive environment, when two teams' performance is exactly the same. However, there are usually many random factors affecting each team's performance, these factors are independent, and the probability that two independent random variables are exactly equal is 0.

Thus, one of the teams will be always slightly ahead, thus providing an incentive for the other team to catch up – and this way, stagnation will be avoided.

6 How to Best Implement This Algorithm

How to best arrange the competition? In the previous text, we simply explained why the competition-related arrangement works well. Now that we know that it works well, a natural next question is: how to best arrange this competition so as to make the education most effective?

There are many aspects to this question: how many teams to select, how exactly to divide students into teams, etc. How better to assign students to teams depends on the affinity between the students. How many situations can be handled within the same time period depends of the students' motivations and stress level, etc. In this section, we deal with only one aspect of this question: into how many teams should we divide the students, and how many students should be in each team. Is it better to divide students into two competing teams – the case we analyzed in the previous text – or is it better to divide them into three or more teams?

For this aspect, it seems reasonable to conclude that, in the first approximation, e.g., the motivations do not change much whether we divide them into two or three or more teams, but what changes a lot is interaction between students, their feelings towards each other – collaboration or competition (and the resulting stress level). So, in this section, for simplicity, we assume that the students' feelings are the main factor that changes when we change the number of teams and the assignment of students to teams. Based on this assumption, we construct yet another (simple) mathematical model of the corresponding phenomenon. We will then use this model to make recommendations about the number of teams and the number of students in each team.

It is important to take student feelings into account. As all teachers know, effectiveness is not the only criterion that we use when selecting a teaching strategy.

We also need to make sure that students feel good about this way of teaching. For example, if we force the students to study 14 hours a day, they will probably learn more – but they will feel stressed and upset.

From the viewpoint of student feelings, which way of dividing the students into teams works better? Let us reformulate this question in precise mathematical terms.

Towards a mathematical model of student feelings. If we divide students into competing teams, then naturally students have:

- positive feeling towards their teammates, but
- somewhat negative feeling towards their competitors – i.e., students from competing teams.

Vice versa, a student gets:

- positive feedback from other students from his/her team and
- negative feedback feeling from students from competing teams.

Since a priori, we do not have any reason to believe that some of these feedbacks are more important than others, it is reasonable to view all these positive and negative feedbacks as equally important. From this viewpoint, for each student:

- each member of his/her team (and this student himself) brings to this student 1 positive feedback unit, and
- each member of each of the competing teams bring to this student 1 negative feedback unit.

If we add up all these units, we conclude that as a natural measure of the student's feelings we can take the difference between the number of other students in this student's team (who bring positive feedback) and the number of students in all competing teams (who bring negative feedback).

Let us describe this in precise terms.

Definitions and the first result.

Definition 1. *Let a positive integer N be given. This integer will be called the* number of students in the class. *The set $S = \{1, \ldots, N\}$ will be called the* set of students*; integers from this set will be called* students.

- *By a* division *into competing teams, we mean a tuple $D = (T_1, \ldots, T_k)$ of non-empty subsets of the set of all students such that the sets T_i are disjoint ($T_j \cap T_{j'} = \emptyset$ when $j \neq j'$), and their union is equal to the whole set S: $T_1 \cup \ldots \cup T_k = S$.*
- *The sets T_j are called* teams.
- *For each student i, let us denote, by $D(i)$, the number j of the team that contains this student: $i \in T_j = T_{D(i)}$.*
- *For each division D and for each student i, by the* feeling $f_i(D)$ *of the i-th student, we mean the value*

$$f_i(D) = (|T_{D(i)}| - 1) - (|T_1| + \ldots + |T_{D(i)-1}| + |T_{D(i)+1}| + \ldots + |T_k|), \qquad (9)$$

where $|T|$ denotes the number of elements in a set T.

- *We say that a division D' is* better *than a division D if for all students i, we have $f_i(D) \le f_i(D')$, and for some students i, we have $f_i(D) < f_i(D')$.*
- *We say that the division D is* optimal *if no other division is better than D.*

Proposition 1. *A division D is optimal if and only if it divides students into two teams $k = 2$.*

Comment. This result explains that, from the viewpoint of student feelings, the best way to organize competition is to divide students into two competing teams.

Proof.

1°. Let us first prove that if any division $D = (T_1, T_2, T_3, \ldots, T_k)$ that divides students into $k > 2$ teams is not optimal.

To prove this, we will show that the division $D' \stackrel{\text{def}}{=} (T_1 \cup T_2, T_3, \ldots, T_k)$ is better than D. Indeed, for all students from the teams T_3 through T_k, the feeling does not change when we go from D to D', but for students from the teams T_1 and T_2 the feelings become better:

- students from team T_1 used to get negative feelings from students from the team T_2 (term $-|T_2|$ in $f_i(D)$), now this feeling is positive, while all other feelings remain unchanged;
- similarly, students from team T_2 used to get negative feelings from students from the team T_1 (term $-|T_1|$ in $f_i(D)$), now this feeling is positive, while all other feelings remain unchanged.

2°. In Part 1 of this proof, we showed that only divisions into two teams can be optimal. To complete the proof, let us show that every division into two teams is optimal.

To prove this, we need to show that no division into two teams can be better that another division into two teams.

Indeed, let us consider a division $D = (T_1, T_2)$ into two teams. Without losing generality, let us assume that the size of the first team is smaller than or equal to the size of the second team. So, this division divides students into a team $n \stackrel{\text{def}}{=} |T_1|$ of size $n \le N/2$ and the remaining team T_2 of size $N - n$. In this division:

- students from the first team get the feeling

$$f_i(D) = (n - 1) - (N - n) = 2n - N - 1 \le -1,$$

while

- students from the second team get the feeling

$$f_i(D) = ((N - n) - 1) - n = N - 2n - 1 \ge -1.$$

Let us consider three possible cases.

2.1°. If two divisions D and D' have the same first-team size n, then they have the same feeling for the same number of students, so the sum of all the feelings is the same – but if D' was better than D, we would have $\sum\limits_{i=1} f_i(D) < \sum\limits_{i=1} f_i(D')$. So, in this case, D cannot be better than D'.

2.2°. Let us now consider the case when the division $D' = (T_1', T_2')$ has more students in its first team than D, i.e., when $n' = |T_1'| > n = |T_1|$. In this case, $N - 2n' < N - 2n$, so $N - 2n' - 1 < N - 2n - 1$. So, every student who had a feeling $N - 2n \geq -1$ in the division D can only get smaller positive feeling in the division D'. So, such D' cannot be better than D.

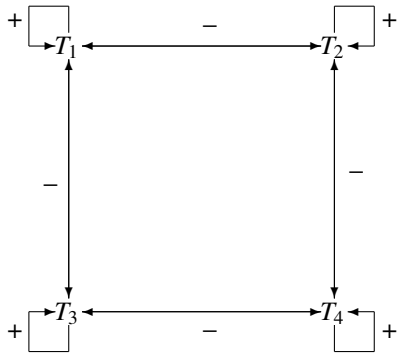
2.3°. Finally, if D' has fewer students in its first team, i.e., if $n' < n$, then we have $2n' - N < 2n - N$, thus $2n' - N - 1 < 2n - N - 1$. So in the division D', some students will get lower negative feelings than in the division D.

2.4°. In all three cases, D' is not better than D.

The statement is proven, and so is the proposition.

Comment. This proposition makes sense: even if we divide the class into more than 2 teams, since the bigger team has an advantage, some team will naturally start cooperating – to increase their chances of succeeding, and we will end up with fewer and fewer competing teams – until it gets to two.

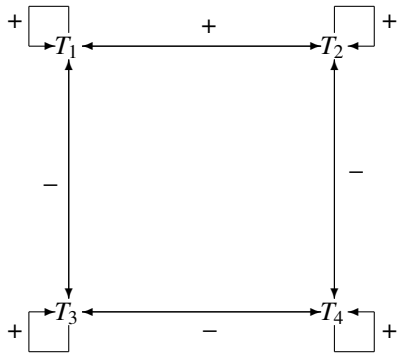
Visual representation of this result. Let us assume that originally, we had four competing teams: T_1, T_2, T_3, and T_4. Students have positive feelings towards their teammates and negative feeling towards students from other teams:



If we merge teams T_1 and T_2 into a single team $T_1 \cup T_2$, then:

- between students of the two original terms T_1 and T_2, the attitude changes from negative to positive, while
- all other feelings remain the same.

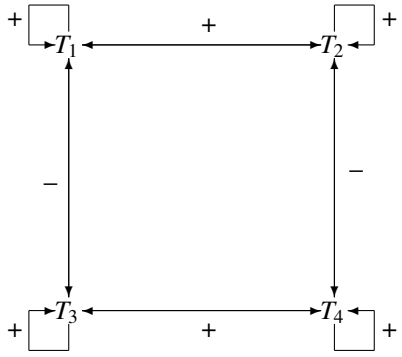
Thus, with this merger, the number of positive connections increases:



Finally, if we merge teams T_3 and T_4 into a single team $T_3 \cup T_4$ – thus leaving only two competing teams: $T_1 \cup T_2$ versus $T_3 \cup T_4$ – then:

- between students of the original terms T_3 and T_4, the attitude changes from negative to positive, while
- all other feelings remain the same.

Thus, with this merger, the number of positive connections increases even further:



So which division into two teams should we select? A natural idea is to select a division in which the worst-case feeling

$$w(D) \stackrel{\text{def}}{=} \min_i f_i(D). \tag{10}$$

is the largest possible. This is described by the following result.

Proposition 2.

- *For even N, the division with the largest possible value of $w(D)$ is the division into two equal size teams.*

- *For odd $N = 2m + 1$, the division with the largest possible value of $w(D)$ is the division into teams of size m and $m + 1$.*

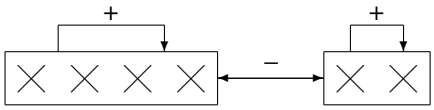
Proof. According to the formulas that we obtained when proving Proposition 1, for any division into teams of size n and $N - n$, the value $w(D)$ is equal to

$$w(D) = \min(2n - N - 1, N - 2n - 1).$$

Since $N - 2n = -(2n - N)$, this means that $w(D) = -|2n - N| - 1$. Thus, the largest possible $w(D)$ corresponds to the smallest possible value of $|2n - N| - 1$.

For even N, the smallest possible value is clearly 0, when $n = N/2$. For $N = 2m+1$, we cannot have 0, but we can have 1, when $n = m$. The proposition is proven.

Visual illustration. Let us illustrate the above idea on a simple example. Suppose that initially, we divided 6 students into two unequal groups: a group of 2 and a group of 4:



In this division, each student from the smaller group has:

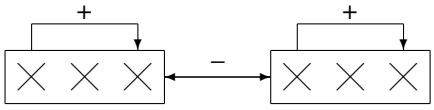
- negative feelings from 4 students from the larger groups and
- positive feelings from his/her partner in the smaller group.

So here $w(D) = 1 - 3 = -2$.

On the other hand, if we instead divide 6 students into two equal groups, then we will get, for each student:

- 3 negative feelings and
- 2 positive feelings.

So here $w(D) = 2 - 3 = -1$m which is larger than -2:



This example shows from the viewpoint of student feelings, the division into equal groups is indeed better.

7 What Next?

What we did. In this paper, we provided a *simplified* mathematical model that explains why adversarial teaching works – and show that, in some reasonable sense, adversarial teaching is indeed a close-to-optimal teaching strategy.

The existence of such an explanation made us (and will hopefully make others) more confident that this method is a right one.

Can we do better? Teaching with more confidence is good, but it would nice to have a model that helps us teach *better*.

For this, we need a more realistic model. Such model should take into account that some attacks are more difficult to defend against, while others are easier. Such models should take into account that in adversarial teaching, it is often not individual against individual, but rather a team against a team – so we need to take into account team dynamics, etc.

We hope that our simplified model will provide a starting point for developing such more realistic models.

How to motivate? This paper presents a mathematical demonstration of how adversarial learning *can be* beneficial for teaching cybersecurity topics. But how to make sure that adversarial learning *is* beneficial?

In this paper, we concentrated on the technical part, on *what* to teach – implicitly assuming that students have the needed motivation (and, of course, the needed background).

In reality, while some students are always eager to learn, but for other students, it is important to keep them motivated. In our experience, when properly organized, competitive environments like hackathons are great motivators – but, on the other hand, pedagogy teaches us that many students do not perform well in competitive environments.

How to best motivate students remains an important open problem.

Acknowledgments

This work was supported in part by the US National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science) and HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are greatly thankful to Boris Kovalerchuk for his encouragement, and to the anonymous referees for valuable suggestions.

References

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., New York, 2009.
2. S. R. Foster, S. Esper, and W. G. Griswold, "From competition to metacognition: designing diverse, sustainable educational games", *Proceedings of the Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems CHI'2013*, Paris, France, April 27 – May 2, 2013, pp. 99–108.
3. W. Gasarch, "The third P=?NP poll", *ACM SIGACT News*, 2019, Vol. 50, No. 1, pp. 40–59.
4. S. T. Hamman and K. M. Hopkinson, "Teaching adversarial thinking for cybersecurity", *Journal of the Colloquium for Information Systems Education (CISSE)*, September 2016, Vol. 4, No. 1.
5. F. Katz, "Adversarial thinking: teaching students to think like a hacker", *Proceedings of the 2019 Kennesaw State University Conference on Cybersecurity Education, Theory and Practice*, Kennesaw, Georgia, October 11–12, 2019.
6. V. Kreinovich and S. A. Starks, "Why benchmarking is an (asymptotically) optimal approach to numerical methods: a geombinatoric proof", *Geombinatorics*, 2004, Vol. 13, No. 3, pp. 131–138.
7. G. G. Lorentz, *Approximation of Functions*, Halt, Reinhart, and Winston, New York, 1966.
8. G. G. Lorentz, "The 13-th problem of Hilbert", in: F. E. Browder (ed.), *Mathematical Developments Arising from Hilbert's Problems*, American Math. Society, Providence, Rhode Island, 1976, Part 2, pp. 419–430.
9. N. Moiseev, *Elements of the Theory of Optimal Systems*, Moscow, Nauka, 1975 (in Russian)
10. C. Servin, O. Kosheleva, and V. Kreinovich, "Adversarial Teaching Approach to Cybersecurity: A Mathematical Model Explains Why It Works Well", *Proceedings of the 24th International Conference on Information Visualisation IV'2020*, Vienna and Melbourne, September 7–11, 2020, pp. 313–316.