Towards an Optimal Design: What Can We Recommend to Elon Musk?

Martine Ceberio, Olga Kosheleva, and Vladik Kreinovich

University of Texas at El Paso, El Paso, Texas 79968, USA
mceberio@utep.edu, olgak@utep.edu, vladik@utep.edu

1. General strategy behind the successes of Elon Musk

- Elon Musk has had many successes in different application areas:
  - from a successful electronic financial system Paypal
  - to practical electric cars
  - to effective reusable rockets for space exploration.
- Recently, a semi-authorized biography of Elon Musk appeared.
- It was semi-authorized in the sense that the author was allowed to follow Musk for several years.
- According to this biography, Musk's general strategy that has led to his many successes is "move fast and break things".

2. General strategy behind Elon Musk's successes (cont-d)

- Musk is motivated by his experience that not all the constraints that are usually required for a design are actually necessary; so:
  - instead of following all the constraints — which would make the design very expensive,
  - he tries to find the smallest number of constraints that are still necessary for the success.
- This minimal necessary number of constraints is usually much smaller than what is currently required.
- As a result, his final design – that does not have to follow all the original constraints – is drastically cheaper.
- Musk uses the fact that:
  - usually, all the constraints can be naturally sorted in the descending order of their importance,
  - so that the most important ones are listed first.

3. General strategy behind Elon Musk's successes (cont-d)

- What Musk does is he guesses the minimally necessary number of constraints, and makes a design based on these many constraints.
- If the resulting design works – e.g., if the rocket successful reaches the orbit – then he tries to see if some other constraints are not necessary.
- Sometimes, the cut was too harsh – and the resulting rocket design is not successful, the rocket explodes.
- Then he tries a design in which more of original constraints are satisfied.
- Of course, Elon Musk was not the first person to use intuition in decision making.
- Use of intuition is a well known practice.
- What was novel in his approach is that he applied intuition to decide how many constraints to take into account.

4. Can we replace intuition in this strategy?

- The above-described strategy is based on intuition, on guessing the minimally necessary number of constraints.
- Sometimes this intuition works, sometimes it does not.
- For example, for a reusable rocket, the first guess, that only 10 constraints are necessary, led to an explosion.
- A natural question is: can we replace intuition with justified recommendations?

5. How can we replace intuition: an idea

- The main objective is to minimize the cost of this search for this minimal number of constraints.
- Usually, Musk deals with a completely new area, in which there is practically no past experience.
- So we cannot come up with meaningful probabilities of different situations.
- In such cases, a natural idea is to minimize the worst-case cost of a strategy.
- We want to find the strategy that minimizes the cost.
- So, we need to take into account that the cost of each experiment drastically depends on whether this experiment
  - led or a success
  - or resulted in a failure.

6. How can we replace intuition: an idea (cont-d)

- For example, if a rocket blows up, the cost is much larger that when it successfully returns to Earth.
- So, we need to distinguish between the cost of success c and the cost of failure C – which is much larger than c.
- We cannot completely avoid failures.
- If we do not experience a failure, how can we be sure that the number of constraints is indeed the smallest possible?
- But what we need to do is to minimize the overall cost of this search, including the cost of possible failures.

7. What is known and what we do in this talk

- To the best of our knowledge, this optimization problem has not been considered earlier.
- In this talk, we present a solution to this optimization problem.
- To come up with this solution, we utilize optimization results about other situations from three of our previous papers:
- In our 2000 paper, we developed an asymptotically optimal algorithm for finding the shortest plan.
- In our 2008 paper, we considered the problem of optimal elicitation of information from an expert, and
- In our 2020 paper, we considered the problem of finding the optimal individual dose of a medicine.

8. What is known and what we do in this talk (cont-d)

- The related practical problems are different from our current problem.
- However, it turns out that from the mathematical and computational viewpoint, all these optimization problems are somewhat similar.
- Thus, our algorithm can be obtained by an appropriate modification of algorithms developed for these previous problems.

9. Comment

- At this moment, what we are proposing is a theoretically justified algorithm.
- This algorithm is based on the assumptions that:
  - we know the order of constraint importance and
  - the known cost estimates are correct.
- In view of these two assumptions, it makes sense to consider our algorithm as the first approximation to the optimal strategy.
- It is desirable to analyze – both theoretically and practically:
  - how to get more accurate strategies,
  - by taking into account that the real-life situation may be somewhat different from these two assumptions.
- Some ordering of constraint importance may be wrong.
- The actual cost estimates may differ from the expert estimates.

10. Problem: reminder

- In general, we have a situation in which:
  - we have a list of N_0 constrains, and
  - we know that following all these constraints leads to a success.
- We also know that if we do not follow any constraints at all, the result will be a failure.
- Our task is to find the number of constraints $k \in (0, N_0)$ to try.
- If the result of this experiment is a failure, this means that the first k constraints are absolutely necessary.
- So the question is how many of the remaining $N_0 - k$ constraints we shall try next.
- If the result of this experiment is a success, this means that we only need some of these k constraints.
- So the question is how many of these k constraints we shall try next.

11. Problem: reminder (cont-d)

- In line with the previous section, we will denote the cost of a successful experiment by c, and the cost of a failed experiment by C.
- The ordering of the constraints by importance and the cost estimates should come from the experts – as in Elon Musk's cases.

12. Let us describe this problem in precise terms

- Let us denote by $c(N)$ the smallest worst-case cost of a strategy that:
  - finds the minimal necessary number of constraints
  - in a situation when we have N original constraints.
- When $N = 1$, there is no need to have any experiments.
- We have only one constraint, and we know that without this constraint, the system will not work.
- In this case, the cost is 0: $c(1) = 0$.
- When $N \geq 2$, we need to find the optimal number of constraints.
- To do that, we select the number of constraints $k \in (0, N)$ to try.
- The resulting cost depends on whether this experiment is a success or a failure.

13. Let us describe this problem in precise terms (cont-d)

- If the experiment of satisfying only $k < N$ constraints was a success, this means that:
  - we spent the amount c on this experiment, and
  - we also need to spend amount $c(k)$ to find the smallest possible number of the selected k constraints.
- So, in this case, the cost if $c + c(k)$.
- If the experiment of satisfying only $k < N$ constraints was a failure, this means that:
  - we spent the amount C on this experiment, and
  - we also need to spend amount $c(N-k)$ to find the smallest possible number of the remaining $N - k$ constraints.
- So, in this case, the cost if $C + c(N - k)$.

14. Let us describe this problem in precise terms (cont-d)

- The worst-case cost of testing k constraints is the largest of these two amounts, i.e.:
$$\max(c + c(k), C + c(N - k)).$$
- The optimal worst-case cost $c(N)$ corresponds to the case when we select k that minimizes this worst-case cost, i.e.:
$$c(N) = \min_{0<k<N} \max(c + c(k), C + c(N - k)).$$
- This formula naturally leads to the following algorithm.

15. Proposed algorithm

- Suppose that we know the values c, C, and N_0.
- Then, we take $c(1) = 0$.
- For $N = 2, 3, \ldots, N_0$, we use the above formula to sequentially compute the values
$$c(2), c(3), \ldots, c(N_0).$$
- In this process, for each value $N \leq N_0$, we get the value $k(N)$ that minimizes the above expression.
- Then, we select $k_0 = k(N_0)$ constraints to try.
- Let us first consider the case when the experiment of satisfying only k_0 constraints was a success.
- This means that we need to find the smallest possible number of the selected k_0 constraints.
- In line with our analysis, we select $k(k_0)$ constraints to try.

16. Proposed algorithm (cont-d)

- Let us now consider the case when the experiment of satisfying only $k_0 < N$ constraints was a failure.
- This means that need to find the smallest possible number of the remaining $N - k_0$ constraints.
- In line with our analysis:
  - we select $k(N - k_0)$ additional constraints to try,
  - in addition to the k_0 constraints that turned out to be absolutely necessary.

17. Numerical example

- Suppose that $c = 1$, $C = 2$, and $N_0 = 4$.
- Here, $c(1) = 0$.
- Then, first, we compute

$$c(2) = \min_{0<k<2} \max(c + c(k), C + c(2-k)) = \max(c + c(1), C + c(1)) =$$

$$\max(1 + 0, 2 + 0) = 2.$$

- In this case, $k(2) = 2$.
- Then, we compute

$$c(3) = \min_{0<k<3} \max(c + c(k), C + c(3-k)) =$$

$$\min(\max(c + c(1), C + c(2)), \max(c + c(2), C + c(1)) =$$

$$\min(\max(1 + 0, 2 + 2), \max(1 + 2, 2 + 0)) = \min(4, 3) = 3.$$

- In this case, the minimum is attained when $k = 2$, so $k(3) = 2$.

18. Numerical example (cont-d)

- After that, we compute

$$c(4) = \min_{0<k<4} \max(c + c(k), C + c(3 - k)) =$$

$$\min(\max(c{+}c(1), C{+}c(3)), \max(c{+}c(2), C{+}c(2)), \max(c{+}c(3), C{+}c(1))) =$$

$$\min(\max(1{+}0, 2{+}3), \max(1{+}2, 2{+}2), \max(1{+}3, 2{+}0)) = \min(5, 4, 4) = 4.$$

- In this case, the minimum is attained when $k = 2$ or when $k = 3$, so we can select both $k(4) = 2$ or $k(4) = 3$.
- If we select $k(4) = 3$, then first, we try satisfying only 3 constraints.

19. Numerical example (cont-d)

- If this results in a success:
  - i.e., if we know that 3 constraints are sufficient,
  - we will then – since $k(3) = 2$ – need to check if two constraints are sufficient.
- If this results in a failure:
  - then we know that 3 constraints are not sufficient,
  - so the original 4 constrains is the smallest number that need to be satisfied.

20. Computational complexity of our algorithm

- For each $N = 2, 3, \ldots, N_0$, to compute the value $c(N)$, we need to compute and compare $N - 1$ sums.
- So the computation time is proportional to $N - 1$.
- Thus, the overall computation time is proportional to

$$(2-1)+(3-1)+\ldots+(N_0-1) = 1+2+\ldots+(N_0-1) = \frac{(N_0-1)\cdot N_0}{2}.$$

- So, the computational complexity is proportional to N_0^2.
- This is quite feasible, since N_0 is usually in the dozens.

21. Asymptotic formula for $c(N)$

- The following result holds:
- **Proposition.** *For every c and C, there exists a constant C_0 such that for all N, we have*
$$|c(N) - a \cdot \log_2(N)| \le C_0.$$
- *Here a is the solution to the equation*
$$2^{-c/a} + 2^{-C/a} = 1.$$
- This result means that asymptotically, for large N, we have
$$c(N) \approx a \cdot \log_2(N).$$
- For example, in the above case $c = 1$ and $C = 2$, the equation for a takes the form $t + t^2 = 1$, where we denoted $t \overset{\text{def}}{=} 2^{-1/a}$.
- Thus, t is the golden ratio
$$t = \frac{\sqrt{5}-1}{2} \approx 0.618 \text{ and } a = -\frac{1}{\log_2(t)}.$$

22. Asymptotic formula for $c(N)$ (cont-d)

- The proof of this proposition is given in the paper.
- This proof also shows that we get an asymptotically optimal strategy if:
  - instead of selecting the optimal $k(N)$,
  - we select the value $k = \lfloor \alpha \cdot N \rfloor$, where $\alpha \overset{\text{def}}{=} 2^{-c/a}$.

23. Proof of the Proposition: General Idea

- Let us consider an alternative procedure B in which:
  - instead of selecting the optimal value $k(N)$,
  - we select the value $k = \lfloor \alpha \cdot N \rfloor$, where $\alpha \stackrel{\text{def}}{=} 2^{-c/a}$.
- Let us denote, by $b(N)$, the worst-case cost of this procedure.
- We will prove that there exist constants $C_0 > 0$ and $C_1 > 0$ such that for every N, we have $a \cdot \log_2(N) \le c(N)$ and

$$b(N) \le a \cdot \log_2(N) + C_0 - \frac{C_1}{N}.$$

- By definition, $c(N)$ is the smallest worst-case cost of all possible procedures; thus, $c(N) \le b(N)$.
- So, if we prove the above two inequalities, we will indeed complete the proof of the Proposition.

24. Proof of the first inequality

- Let us first prove the first inequality by induction over N.
- The value $N = 1$ represents the induction base.
- For this value, $a \cdot \log_2(1) = 0 = c(1)$, so the inequality holds.
- Let us now describe the induction step.
- Suppose that we have already proved the inequality $a \cdot \log_2(n) \le c(n)$ for all $n < N$.
- Let us prove that $a \cdot \log_2(N) \le c(N)$.
- By definition $c(N)$ is the smallest of the values

$$\max\{c + c(k), C + c(N-k)\} \text{ over } k = 1, 2, \ldots, N-1.$$

- We want to prove that $a \cdot \log_2(N)$ is indeed the lower bound for $c(N)$.

25. Proof of the first inequality (cont-d)

- So, we must prove:
  - that $a \cdot \log_2(N)$ cannot exceed each of these values,
  - i.e., that $a \cdot \log_2(N) \le \max\{c + c(k), C + c(N-k)\}$ for every $k = 1, 2, \ldots, N-1$.
- For these k, we have $k < N$ and $N - k < N$.
- So, for all these values, we already know that $a \cdot \log_2(k) \le c(k)$ and
$$a \cdot \log_2(N-k) \le c(N-k).$$
- Therefore, $c + a \cdot \log_2(k) \le c + c(k)$,
$$C + a \cdot \log_2(N-k) \le C + c(N-k), \text{ and}$$
$$\max\{c+a\cdot\log_2(k), C+a\cdot\log_2(N-k)\} \le \max\{c+c(k), C+c(N-k)\}.$$

26. Proof of the first inequality (cont-d)

- So, to prove the desired inequality, it is sufficient to prove that

$$a \cdot \log_2(N) \le \max\{c + a \cdot \log_2(k), C + a \cdot \log_2(N - k)\}.$$

- We will prove this inequality by considering two possible cases:

$$k \le \alpha \cdot N \text{ and } k \ge \alpha \cdot N.$$

27. Case when $k \le \alpha \cdot N$

- When $k \le \alpha \cdot N$, we have $N - k \ge (1 - \alpha) \cdot N$ and therefore,

$$C + a \cdot \log_2(N - k) \ge z, \text{ where}$$

$$z \stackrel{\text{def}}{=} C + a \cdot \log_2((1 - \alpha) \cdot N) = C + a \cdot \log_2(N) + a \cdot \log_2(1 - \alpha).$$

- Here, by definition of α, we have $\alpha = 2^{-c/a}$, and by definition of a, we have $2^{-c/a} + 2^{-C/a} = 1$. Thus, $1 - \alpha = 2^{-C/a}$.
- Hence $\log_2(1 - \alpha) = -C/a$, so $C + a \cdot \log_2(1 - \alpha) = 0$.
- Thus, $z = a \cdot \log_2(N)$. In this case,

$$a \cdot \log_2(N) \le z = C + a \cdot \log_2(N - k) \le \max\{c + a \cdot \log_2(k), C + a \cdot \log_2(N - k)\}.$$

28. Case when $k \ge \alpha \cdot N$

- When $k \ge \alpha \cdot N$, we have $c + a \cdot \log_2(k) \ge z$, where

$$z \overset{\text{def}}{=} c + a \cdot \log_2(\alpha \cdot N) = c + a \cdot \log_2(N) + a \cdot \log_2(\alpha).$$

- By definition of α, we have $\alpha = 2^{-c/a}$, hence $\log_2(\alpha) = -c/a$, and $z = a \cdot \log_2(N)$.
- So, in this case,

$$a \cdot \log_2(N) \le z = c + a \cdot \log_2(k) \le \max\{c + a \cdot \log_2(k), C + a \cdot \log_2(N - k)\}.$$

- In both cases, we have the desired inequality.
- The induction step is proven.
- So, indeed, for every N, we have $a \cdot \log_2(N) \le c(N)$.

29. Proof of the second inequality

- Let us now prove that there exist real numbers $C_0 > 0$ and $C_1 > 0$ for which, for all N,

$$b(N) \leq a \cdot \log_2(N) + C_0 - \frac{C_1}{N}.$$

- To prove this inequality, we will:
  - pick a value N_0,
  - prove that this inequality holds for all $N \leq N_0$, and then
  - use mathematical induction to show that it holds for all $N > N_0$ as well.

30. Induction basis

- Let us first find the conditions on C, C_1, and N_0 under which for all $N \le N_0$,
$$b(N) \le a \cdot \log_2(N) + C_0 - \frac{C_1}{N}.$$
- Subtracting $a \cdot \log_2(N)$ and adding $\frac{C_1}{N}$ to both sides of the this inequality, we get
$$C_0 \ge \frac{C_1}{N} + b(N) - a \cdot \log_2(N) \text{ for all } N \text{ from } 1 \text{ to } N_0.$$
- So, to guarantee that this inequality holds, if we have already chosen N_0 and C_1, we can choose
$$C_0 = \max_{1 \le N \le N_0} \left(\frac{C_1}{N} + b(N) - a \cdot \log_2(N) \right).$$

31. Induction step

- Let us assume that for all $n < N$ (where $N > N_0$), we have proven that
$$b(n) \le a \cdot \log_2(n) + C_0 - \frac{C_1}{n}.$$
- We would like to conclude that
$$b(N) \le a \cdot \log_2(N) + C_0 - \frac{C_1}{N}.$$
- According to the definition of $b(N)$, we have
$$b(N) = \max\{c + b(k), C + b(N-k)\}, \text{ where } k = \lfloor \alpha \cdot N \rfloor.$$
- Due to induction hypothesis, we have
$$b(k) \le a \cdot \log_2(k) + C_0 - \frac{C_1}{k} \text{ and}$$
$$b(N-k) \le a \cdot \log_2(N-k) + C_0 - \frac{C_1}{N-k}.$$

32. Induction step (cont-d)

- Therefore,

$$b(N) \le \max\left\{c + a\cdot\log_2(k) + C_0 - \frac{C_1}{k},\ C + a\cdot\log_2(N-k) + C_0 - \frac{C_1}{N-k}\right\}.$$

- Thus, to complete the proof, it is sufficient to conclude that this maximum does not exceed

$$a\cdot\log_2(N) + C_0 - \frac{C_1}{N}.$$

- In other words, we must prove:

$$c + a\cdot\log_2(k) + C_0 - \frac{C_1}{k} \le a\cdot\log_2(N) + C_0 - \frac{C_1}{N} \text{ and}$$

$$C + a\cdot\log_2(N-k) + C_0 - \frac{C_1}{N-k} \le a\cdot\log_2(N) + C_0 - \frac{C_1}{N}.$$

- Without losing generality, let us show how we can prove the first of these two inequalities.

33. Induction step (cont-d)

- Since $k = \lfloor \alpha \cdot N \rfloor$, the left-hand side of the inequality can be rewritten as

$$c + a \cdot \log_2(\alpha \cdot N) + a \cdot (\log_2(k) - \log_2(\alpha \cdot N)) + C_0 - \frac{C_1}{k}.$$

- We already know that $c + a \cdot \log_2(\alpha \cdot N) = a \cdot \log_2(N)$.
- Thus, the left-hand side takes the simpler form

$$a \cdot \log_2(N) + a \cdot (\log_2(k) - \log_2(\alpha \cdot N)) + C_0 - \frac{C_1}{k}.$$

- Let us substitute this expression into the formula and cancel the terms $a \cdot \log_2(N)$ and C_0 in both sides.
- Then, we get an equivalent inequality

$$a \cdot (\log_2(k) - \log_2(\alpha \cdot N)) - \frac{C_1}{k} \le -\frac{C_1}{N}.$$

- Let us further simplify this inequality.

34. Induction step (cont-d)

- We will start by estimating the difference $\log_2(k) - \log_2(\alpha \cdot N)$.
- To estimate this difference, we will use the intermediate value theorem:
  - for every smooth function $f(x)$, and for arbitrary two values A and B,
  - we have $f(A) - f(B) = (A - B) \cdot f'(\xi)$ for some $\xi \in [A, B]$.
- In our case, $f(x) = \log_2(x) = \dfrac{\ln(x)}{\ln(2)}$, $A = k$, and $B = \alpha \cdot N$.
- Here, $f'(\xi) = \dfrac{1}{\xi \cdot \ln(2)}$, so $f'(\xi) \le \dfrac{1}{k \cdot \ln(2)}$.
- Also, $|A - B| \le 1$, so, the difference $\log_2(k) - \log_2(\alpha \cdot N)$ can be estimated from above by:

$$\log_2(k) - \log_2(\alpha \cdot N) \le \frac{1}{k \cdot \ln(2)}.$$

35. Induction step (cont-d)

- Hence, the above inequality holds if the following stronger inequality holds:
$$\frac{a}{k \cdot \ln(2)} - \frac{C_1}{k} \le -\frac{C_1}{N}.$$
- This desired inequality is equivalent to
$$\frac{C_1}{N} \le \frac{C_1 - a/\ln(2)}{k}.$$
- Here, $k \ge \alpha \cdot N - 1$, i.e.,
$$\frac{k}{N} \ge \alpha - \frac{1}{k}.$$
- When $N \to \infty$, we have $k \to \infty$ and $\dfrac{1}{k} \to 0$.
- Thus, for every $\varepsilon > 0$, there exists an N_0 starting from which $\dfrac{1}{k} \le \varepsilon$ and hence, $k \ge (\alpha - \varepsilon) \cdot N$.

36. Induction step (cont-d)

- For such sufficiently large N, the desired inequality can be proven if we have
$$\frac{C_1}{N} \le \frac{C_1 - a/\ln(2)}{(\alpha - \varepsilon) \cdot N}.$$
- This is equivalent to
$$C_1 \le \frac{C_1 - a/\ln(2)}{\alpha - \varepsilon}.$$
- Since $\alpha \le 1$, for sufficiently large C_1, this inequality is true.
- For such C_1, therefore, the induction can be proven and thus, the Proposition is proven.
- We have also proved – as promised – that the Algorithm B leads to asymptotically optimal worst-case cost.

37. Acknowledgments

This work was supported in part by:

- National Science Foundation grants 1623190, HRD-1834620, HRD-2034030, and EAR-2225395;
- AT&T Fellowship in Information Technology;
- program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and
- a grant from the Hungarian National Research, Development and Innovation Office (NRDI).