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

1st Martine Ceberio
Department of Computer Science
University of Texas at El Paso
El Paso, Texas, USA
mceberio@utep.edu

2nd Olga Kosheleva
Department of Teacher Education
University of Texas at El Paso
El Paso, Texas, USA
olgak@utep.edu

3rd Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, Texas, USA
vladik@utep.edu

4th Hung T. Nguyen
Department of Mathematical Science
New Mexico State University
Las Cruces, New Mexico, USA, and
Faculty of Economics
Chiang Mai University
Chiang Mai, Thailand
hunguyen@nmsu.edu

***Abstract*—Elon Musk's successful "move fast and break things" strategy is based on the fact that in many cases, we do not need to satisfy all usual constraints to be successful. By sequentially trying smaller number of constraints, he finds the smallest number of constraints that are still needed to succeed – and using this smaller number of constrains leads to a much cheaper (and thus, more practical) design. In this strategy, Musk relies on his intuition – which, as all intuitions, sometimes works and sometimes doesn't. To replace this intuition, we propose an algorithm that minimizes the worst-case cost of finding the smallest number of constraints.**

***Index Terms*—optimal design, worst-case cost, move fast and break things, Elon Musk**

I. Formulation of the problem

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.

According to a recent semi-authorized biography of Elon Musk [2] – semi-authorized in the sense that the author was allowed to follow Musk for several years – Musk's general "move fast and break things" strategy (that has led to his many successes) is motivated by his experience that not all the constraints that are usually required for a design are actually necessary. 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's usual strategy for finding the minimal number of constraints 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. 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.
- If the cut was too harsh – and the resulting rocket design is not successful, the rocket explodes – then he tries a design in which more original constraints are satisfied.

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?

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.

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology. It was also supported by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

In finding the strategy that minimizes the cost, we need to take into account that the cost of each experiment drastically depends on whether this experiment led or a success or to a failure: 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.

What is known and what we do in this paper. In this paper, we present a solution to the corresponding optimization problem.

To come up with this solution, we utilize optimization results about other situations from three of our previous papers:

- in [5], [6], we developed an asymptotically optimal algorithm for finding the shortest plan;
- in [4] (see also [3]), we considered the problem of optimal elicitation of information from an expert, and
- in [1], we considered the problem of finding the optimal individual dose of a medicine, in a typical situation in which the overdose of the medicine may be much more harmful than its underdose.

While the related practical problems are different from our current problem, 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.

II. Analysis of the Problem and the Resulting Optimal Algorithm

Problem: reminder. In general, we have a situation in which we have a list of N 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 $k \in (0, N)$ 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 $N - k$ 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.

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.

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:

- 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 $C + c(N - k)$.

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.:

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

This formula naturally leads to the following algorithm.

Proposed algorithm. Suppose that we know the values c, C, and N_0. Then, we take $c(1) = 0$, and for $N = 2, 3, \ldots, N_0$, we use the formula (1) 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 expression (1).

Then, we select $k_0 = k(N_0)$ constraints to try.

- If 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.
- If 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.

Numerical example. Suppose that $c = 1$, $C = 2$, and $N_0 = 4$. Here, $c(1)$. 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 $k(3) = 2$.

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. Then:

- 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.

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 – which is quite feasible, since N_0 is usually in the dozens.

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

$$|c(N) - a \cdot \log_2(N)| \le C_0,$$

where a is the solution to the equation

$$2^{-c/a} + 2^{-C/a} = 1.$$

Comments.

- 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
$$t = \frac{\sqrt{5} - 1}{2} \approx 0.618,$$
and
$$a = -\frac{1}{\log_2(t)}.$$
- The proof of this proposition is given in the Appendix.
- 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 $\alpha \overset{\text{def}}{=} 2^{-c/a}$.

References

[1] M. Ceberio, O. Kosheleva, and V. Kreinovich, "Optimal search under constraints", *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society NAFIPS'2020*, Redmond, Washington, August 20-22, 2020, pp. 421–426.

[2] W. Isaacson, *Elon Musk*, Simon & Schuster, New York, 2023.

[3] E. N. Kamoroff, *How to Extract Knowledge From an Expert so That His Effort is Minimal*, Master Project, Computer Science Department, University of Texas at El Paso, 1993.

[4] H. T. Nguyen, V. Kreinovich, and E. Kamoroff, "Asymmetric information measures: how to extract knowledge from an expert so that the expert's effort is minimal", *International Journal of Automation and Control (IJAAC)*, 2008, Vol. 2, No. 2/3, pp. 153–177.

[5] R. A. Trejo, J. Galloway, C. Sachar, V. Kreinovich, C. Baral, and L. C. Tuan, "From planning to searching for the shortest plan: an optimal transition", *Proceedings of the International Conference on Intelligent Technologies*, Bangkok, Thailand, December 13–15, 2000, pp. 17–23.

[6] R. A. Trejo, J. Galloway, C. Sachar, V. Kreinovich, C. Baral, and L. C. Tuan, "From planning to searching for the shortest plan: an optimal transition", *International Journal of Uncertainty, Fuzziness, Knowledge-Based Systems (IJUFKS)*, 2001, Vol. 9, No. 6, pp. 827–838.

Appendix: Proof of the Proposition

General idea. To prove this result, 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 $\alpha \overset{\text{def}}{=} 2^{-c/a}$. 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.

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

$$\max\{c + c(k), C + c(N-k)\}$$

over $k = 1, 2, \ldots, N-1$. So, to prove that $a \cdot \log_2(N)$ is indeed the lower bound for $c(N)$, 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,

$$c + a \cdot \log_2(k) \le c + c(k),$$

$$C + a \cdot \log_2(N-k) \le C + c(N-k),$$

and

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

$$\max\{c + c(k), C + c(N-k)\}.$$

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: $k \le \alpha \cdot N$ and $k \ge \alpha \cdot N$:

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

  $$C + a \cdot \log_2(N-k) \ge z,$$

  where

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

  $$C + a \cdot \log_2(N) + a \cdot \log_2(1-\alpha).$$

  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

  $$C + a \cdot \log_2(1-\alpha) = 0,$$

  and thus, $z = a \cdot \log_2(N)$. In this case,

  $$a \cdot \log_2(N) \le z = C + a \cdot \log_2(N-k) \le$$

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

- 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, and so, indeed, for every N, we have

$$a \cdot \log_2(N) \le c(N).$$

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,

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

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

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

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

for all N from 1 to N_0. So, to guarantee that this inequality holds, if we have already chosen N_0 and C_1, we can choose

$$C_0 = \max_{1 \le N \le N_0} \left(\frac{C_1}{N} + b(N) - a \cdot \log_2(N) \right).$$

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

$$b(N) = \max\{c + b(k), C + b(N-k)\},$$

where $k = \lfloor \alpha \cdot N \rfloor$. Due to induction hypothesis, we have

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

and

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

Therefore,

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

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

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 that

$$c + a\cdot\log_2(k) + C_0 - \frac{C_1}{k} \le a\cdot\log_2(N) + C_0 - \frac{C_1}{N} \quad (2)$$

and that

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

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

Without losing generality, let us show how we can prove the first of these two inequalities. Since $k = \lfloor \alpha\cdot N\rfloor$, the left-hand side of the inequality (2) 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 of (2) takes the simpler form

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

Substituting this expression into (2) and canceling the terms

$$a\cdot\log_2(N)$$

and C_0 in both sides, we get an equivalent inequality

$$a\cdot(\log_2(k) - \log_2(\alpha\cdot N)) - \frac{C_1}{k} \le -\frac{C_1}{N}. \quad (3)$$

Let us further simplify this inequality. 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, according to which, 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,

$$f(x) = \log_2(x) = \frac{\ln(x)}{\ln(2)},$$

$A = k$, and $B = \alpha\cdot N$. Here,

$$f'(\xi) = \frac{1}{\xi\cdot\ln(2)},$$

so

$$f'(\xi) \le \frac{1}{k\cdot\ln(2)};$$

also, $|A - B| \le 1$, so, the difference $\log_2(k) - \log_2(\alpha\cdot N)$ can be estimated from above by:

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

Hence, the above inequality holds if the following stronger inequality holds:

$$\frac{a}{k\cdot\ln(2)} - \frac{C_1}{k} \le -\frac{C_1}{N},$$

or, equivalently,

$$\frac{C_1}{N} \le \frac{C_1 - a/\ln(2)}{k}. \quad (4)$$

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, $k \ge (\alpha - \varepsilon)\cdot N$. For such sufficiently large N, the inequality (4) can be proven if we have

$$\frac{C_1}{N} \le \frac{C_1 - a/\ln(2)}{(\alpha-\varepsilon)\cdot N},$$

i.e., if we have

$$C_1 \le \frac{C_1 - a/\ln(2)}{\alpha - \varepsilon}. \quad (5)$$

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.