Self-Crashing Cars

Edgar G. Daylight

13 September 2019

Abstract

I present a thought experiment on self-driving cars, based on my experience as a safety engineer in the European automotive industry. My main concern is that autonomous vehicles could prove to be disastrous once they reach wide enough deployment; that is, once they reach a critical mass in society, thereby providing the precondition for novel catastrophic events. A compelling research contribution would be to give some helpful (even if speculative) data, probabilities, or estimates, on the comparison between human accidents and accidents caused by autonomous cars. Attempts to do so follow, including probabilistic findings and technical guidelines for engineers and policymakers. Any feedback and cooperation is welcome, coming from historians and philosophers of computing.

1 Brief Intro

My consultancy work in safety engineering has led to multiple concerns about the deployment of self-driving cars that are supposed to be safer than the vehicles we use today. My most worrisome hazard in this regard is the following:

> A seemingly insignificant software error in a network-connected, automated car, can — when detected by a malicious hacker — result in a crash of that car and in thousands of other crashes *due to network connectivity.*

Intuitively, autonomous vehicles could prove to be disastrous once they reach a critical mass in society, thereby providing the precondition for novel catastrophic events. I present a Pareto-based analysis in order to give some helpful (even if speculative) probabilities on the comparison between human accidents and accidents caused by autonomous cars. I end with technical guidelines for engineers and policymakers. The guidelines are not unprecedented, but to the best of my knowledge the Pareto-based though experiment leading to the guidelines is.

2 A Pareto Plot of Cars

Figure 1 presents a Pareto plot: a collection of three Pareto points, P, Q, and R, with each point representing a digital system, consisting of software and

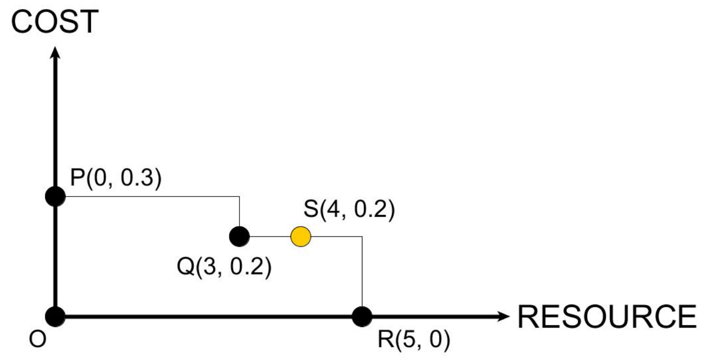

Figure 1: Pareto points P, Q, and R, representing a self-driving car, a conventional car, and an oldtimer, respectively. The resource axis varies from completely automatic (O) to fully manual (R) control of a vehicle. The cost axis is a probability (varying from 0 to 1), capturing the life-threatening manifestations of software/hardware errors in a vehicle.

hardware. The Pareto plot depicts a global trade-off between the amount of resources used (horizontal axis) and the associated cost (vertical axis). More resources are needed to reduce the cost.

The resource axis pertains to a human-driving resource: system $Q(3,\ 0.2)$ of a conventional car requires *more* input from the human driver than system $P(0,\ 0.3)$ of a self-driving car and that is why Q is to the right of P. In other words, Q is *less* constrained than P. Likewise, an oldtimer R requires the human driver to fulfill more tasks than the driver of Q.

The cost in Figure 1 is the probability p, with $0 \leq p \leq 1$, that a software error or hardware bug will contribute to injury or death of a human during the first 100 years of the system's deployment. The contribution can be indirect in the following sense: a malicious hacker exploits a seemingly insignificant software error in the digital system P of a self-driving car, the hacker then succeeds in taking over control of P, along with other cars that are connected to P's network — provided that P has network connectivity.

Figure 1 can be further clarified as follows. $Q(3,\ 0.2)$ is the digital system of a conventional car with a cost of 0.2, while an oldtimer, $R(5,\ 0)$, has zero cost because it has no software nor hardware. Point $S(4,\ 0.2)$ represents a suboptimal digital system — i.e., it is outperformed in resource consumption by $Q(3,\ 0.2)$ and in cost by $R(5,\ 0)$ — and, therefore, S is *not* a Pareto point.

No digital system is perfect and, therefore, neither is digital system P of a self-driving car: the cost of P is strictly larger than zero: $cost(P) = 0.3 > 0$. A conventional car has more digital components than an oldtimer, but much less than a self-driving car. So, intuitively, one would expect the cost of Q to lie in between that of P and R. If this assumption does not hold, then we end up with two Pareto points, P and R, instead of three. Regardless of whether the assumption holds, no claim is made so far that conventional cars or oldtimers will save more lives than self-driving cars. (The prevailing opinion today is actually the opposite.)

What matters now is whether Figure 1 is representative for a thought experiment about automated vehicles. Specifically, does it make sense to talk only about Pareto plots that contain at least two Pareto points? What does a plot of precisely one Pareto point entail in the present context? The answer is simple: one Pareto point laying on the horizontal axis and to the left of R. Particularly, the origin O depicts the *perfect* digital system of a fully automated car, containing not a single engineering mistake. All other points are suboptimal with respect to O. The holy grail of computer science is, after all, to engineer systems that are both correct by construction and fully automated.

In this paper I shall realistically assume that software engineering is about making trade-offs: all Pareto plots of interest have two or more Pareto points. This paper can thus be viewed as a refinement of remarks made by Dietterich and Horvitz concerning bugs and cybersecurity [2] but my recommendation to eschew network-connected, automated cars is a much more critical viewpoint on the topic at hand.

Later on I will also peruse the Pareto plot in Figure 2 in which the resource axis now denotes the degree of automation: from manual control (far left) to complete automation (far right). The cost, in turn, is the probability q, with $0 \leq q \leq 1$, that a human-driving error will contribute to injury or death of a human during the first 100 years since the driver has acquired a driver's license. Again, points P, Q, and R denote systems of a self-driving car, conventional car, and an oldtimer, respectively. Here, however, the self-driving car P scores optimal in terms of cost and the oldtimer R scores the worst of all.[1]

Although a thought experiment lies at the heart of this paper, it should perhaps be mentioned that even a more realistic variant of Figure 2 will, just like Figure 2 itself, not contain many Pareto points in between Q and P. Casner et al. have observed that it is difficult to get drivers who are out of the role of active control back in the control loop when they are needed [1]. Casner is very skeptical about vehicles that are substantially more automated than Q yet are not fully automated and thus unlike P.

3 Comparing Pareto Plots

Sticking to Figure 1 for now, suppose that all vehicles P, Q, and R have the same maximum speed m. What happens if we increase the maximum speed m for each vehicle to M? We obtain a new Pareto plot, depicted in Figure 3, with corresponding Pareto points P', Q', and R', respectively. Observe that point R' coincides with point R. Furthermore, both P' and Q' are located higher than P and Q, respectively. It is fortunately not necessary to address the question whether $cost(P') - cost(P)$ is larger, smaller, or equal to $cost(Q') - cost(Q)$.

[1] A more elaborate exposition could discuss Pareto plots in which the resource axis denotes network connectivity and where "privacy" is another cost to be considered.

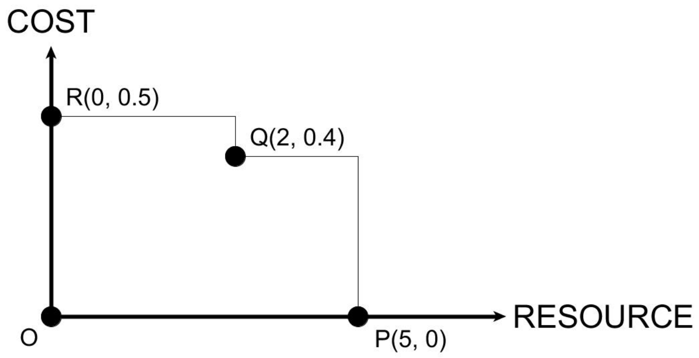

Figure 2: A Pareto plot in which the resource axis now varies from fully manual (O) to completely automatic (P) control of a vehicle. The cost axis is a probability that captures the life-threatening manifestations of human-driving errors.

Another way to compare Figure 3 with Figure 1 is to assume that vehicles P', Q', and R' are engineered to drive on arbitrarily chosen roads in, say, Paris, while vehicles P, Q, and R can only drive on predetermined Parisian roads (or perhaps *rail*roads only). Clearly, the latter scenario is preferable from a safety perspective and this preference is partially captured by the lower cost values of P, Q, R in Figure 1, compared to those shown for P', Q', R' in Figure 3.

Yet another way to compare Figures 1 and 3 is to assume that all vehicles P, Q, and R have *no* network connectivity. (This means that self-driving car P is able to function autonomously without reliance on other communicating vehicles and Internet services.) By providing the same network connectivity to P, Q, and R, we obtain vehicles P', Q', and R', respectively. Again, $R = R'$ because oldtimers cannot use a network. Is it fair to say that P' and Q' will lay above P and Q, respectively? Loosely speaking, does an increase in network connectivity result in more life-threatening manifestations of software/hardware errors (even though it very likely leads to less errors *tout court*)? Again, this is a non-trivial question which fortunately need not be addressed head on.

4 Combining Pareto Plots

The previous discussion is too simplistic — and, therefore, the questions raised are not very significant — for we have focused solely on one vehicle, isolated from other vehicles, bicyclists, pedestrians, etc. Let's contemplate a Pareto plot in which each Pareto point denotes a *composite* digital system: a system of one car, along with that of another car — assuming, of course, that both cars are connected in some sense.

Three broadly construed forms of car *connectivity* are the following, from weak to strong connectivity:

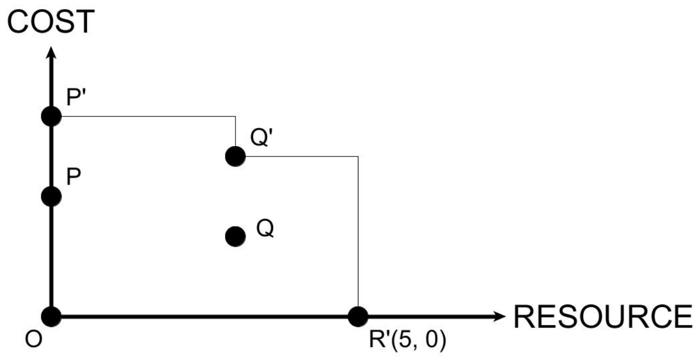

Figure 3: Pareto points P', Q', and R', representing, for example, cars that have a greater maximum speed than those represented by P, Q, and R (with $R = R'$).

1. Two oldtimers passing each other at an intersection.
2. Two cars communicating with each other over the Internet, but both cars do not drive the same roads.
3. A hybrid example: two cars communicating over a local area network while simultaneously passing the same intersection.

To keep our thought experiment simple, consider two oldtimers $R(5, 0)$ and $\hat{R}(5, 0)$ and two self-driving cars $P(0, 0.3)$ and $\hat{P}(0, 0.4)$ with the latter two connected over a network. The Pareto plots are shown in Figure 4(a) and 4(b). Now take each possible combination of two cars, with one car coming from Figure 4(a) and the other one from Figure 4(b).[2] For example, $P \parallel \hat{R}$ represents a composite system in which self-driving car P can sense the presence of car $\hat{R}$. Likewise, $P \parallel \hat{P}$ mathematically captures, amongst other things, the network communication between self-driving cars P and $\hat{P}$.

Figure 4(c) presents the Pareto plot of the aforementioned composite digital systems. Two remarks can be made. First, the cost of $R \parallel \hat{R}$ remains zero simply because the composite system $R \parallel \hat{R}$ contains no software nor hardware. Second, even though four Pareto points are shown, what matters here is that we have at least two Pareto points with the leftmost Pareto point ($P \parallel \hat{P}$) on the vertical axis having a much larger cost than that of P and $\hat{P}$ in isolation. Indeed, connecting two cars P and $\hat{P}$ leads to $cost(P \parallel \hat{P})$ which has the following lower bound: $1 - (1 - cost(P)) \cdot \left(1 - cost(\hat{P})\right) = 0.58$. More generally:

> (A.) Take n cars, none of which is an oldtimer. Then we may realistically assume that the cost p_i of each car is strictly larger than

[2]The reader can freely replace Figure 4(b) with Figure 4(a) or with a plot containing several more Pareto points and then conduct a similar thought experiment.

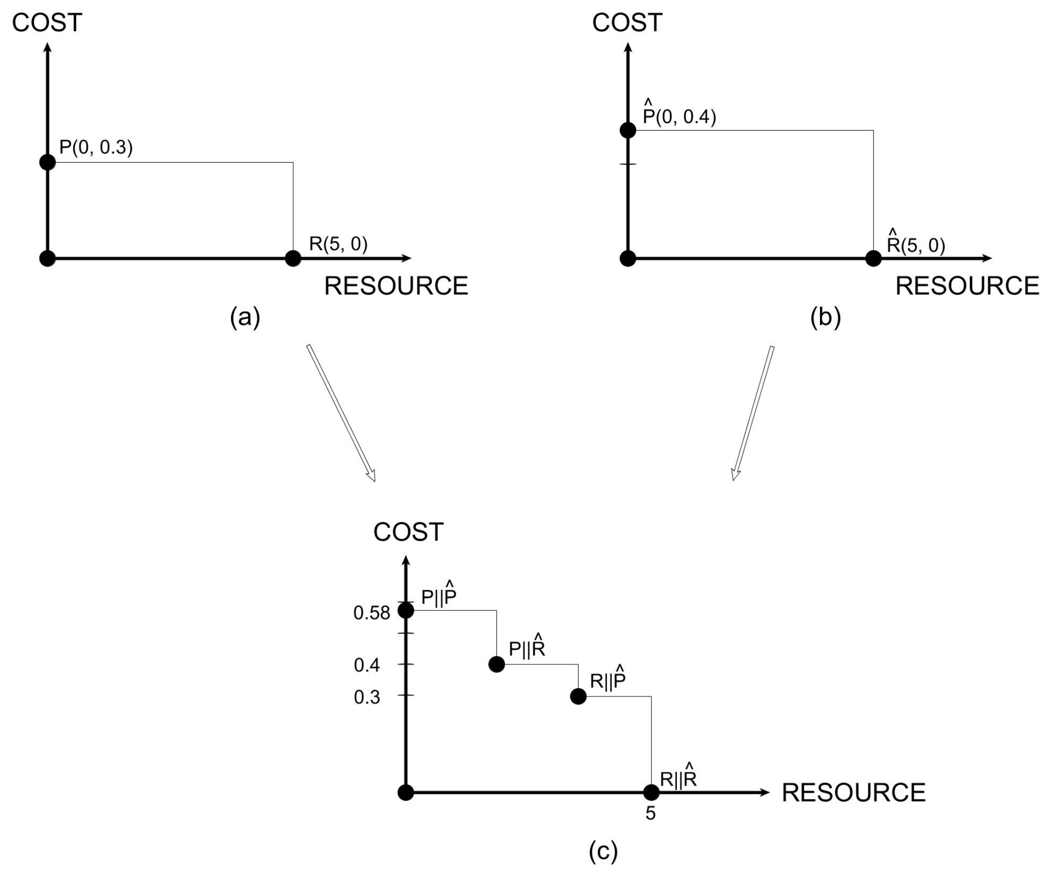

Figure 4: Combining two Pareto plots (a) and (b) into a new Pareto plot (c) in which each point represents a composite system.

> 0. Combining these n cars leads to a composite cost bounded from below by $1 - \prod_{i=1}^{n} (1 - p_i)^n$, and, in the limit ($n \mapsto \infty$), by 1.

In layman's terms: if we deploy more and more cars that contain some software (Q) or a lot of software (P), then the probability that somebody gets injured or killed due to a software/hardware error reaches 1. A visual aid can be obtained by combining Figure 4(c) with Figure 4(a) or Figure 4(b), and continuing aggregating in this way $n - 2$ times. The result is then a Pareto plot with each point $X_1 \parallel X_2 \parallel \ldots \parallel X_n$ denoting a composite digital system for n vehicles. Most importantly, the line segment connecting the leftmost and rightmost Pareto points will have a steep descent, and, in the limit, go from $(0, 1)$ to $(5, 0)$.

However, as it stands, the obtained observation is rather lame for the following is of course also true:

> (B.) If we deploy more and more cars that require some or a lot of manual control, then the probability that a human-driving error leads to injury or death of somebody reaches 1 in the limit.

A visual aid of this last statement can be obtained by combining Figure 2 with Figure 2 and continuing aggregating in this way $n - 1$ times. Based on a similar mathematical treatment as presented above, the two outermost Pareto points will specify a line segment with a steep slope.

My analysis (B.) about human-driving errors is, relative to my analysis about digital errors (A.), more accurate for in the latter case I have yet to mathematically model the synergistic effect that network connectivity between n automated cars has on the cost. Specifically, and as is well-known in safety engineering:

> "If you connect everything together a small mistake in a trivial piece of software can imaginably produce a catastrophe in a connected critical system." — citing Michael Jackson [4, p.87]

From a cybersecurity perspective, a malicious hacker will break into the weakest car in order to steal or crash several, if not all, network-connected cars.

Consider then the following two extremely different worlds:

- World I of 1000 oldtimers, each having a probability p_1 that its driver will cause an accident leading to his injury or death.
- World II of 1000 network-connected, automated cars, which, for simplicity's sake, we divide into 10 mutually disjoint networks (i.e., each network connects 100 cars). Probability p_{100} is the chance that a network is hacked and completely compromised in the sense that all 100 cars in the network injure or kill one of their passengers.

In World I and World II the expected number of non-crashing cars amounts to $1000 - 1000 \times p_1$ and $(10 - 10 \times p_{100}) \times 100$, respectively. So, World I is safer than World II if and only if $p_1 < p_{100}$. In other words:

> We should only prefer a world of network-connected, automated cars if the probability p_{100} that a network is severely compromised is lower than the probability p_1 that a human makes a harmful mistake while driving an oldtimer.

Notice that, just like p_1, probability p_{100} symbolizes *human imperfection*, namely the impossibility to build correct-by-construction digital systems.

In this paper I take the word "correct" to also entail "secure" because a necessary condition for correct software (broadly construed) is that it meets all requirements, including those pertaining to security. A list of excellent books on correctness and reliability include *Future Crimes: Inside the Digital Underground and the Battle for our Connected World [3], Engineering a Safer World [5]*, and *Mechanizing Proof [6]*. Following these books, I not only share recently voiced concerns about car hacking threats [7, 8] and cybersecurity in general [9], I also stress that the auto industry — or any industry of cyber-physical systems that engage in network activity (if not Internet activity) — is *principally* unable to get security *completely* right. Therefore, eschewing networks is one of the best things we can do for driving, but of course not for other tasks such as emailing.

5 Pareto-Based Guidelines

Based on the previous open-ended discussion, a seemingly feasible remedy then is to invest much more effort in getting *each* subsystem X_i, $1 \leq i \leq n$, as correct as possible so that the composite system $X_1 \parallel X_2 \parallel \ldots \parallel X_n$ has a much lower cost than the one discussed until now. Unfortunately, Pareto points P_i already represent, by their very definition, the *optimal* digital systems that the current state of the art can provide us.

Unless we actually find the holy grail of computer science, the best we can do in the interest of safety is adhere to one or more common-sense guidelines, such as those presented in the following non-exhaustive list:

- Mechanically and drastically reduce each vehicle's maximum speed. An Internet of Moving Things is after all intrinsically less safe than an Internet of Static Things (although some static things, such as Internet-connected ovens that may burn kitchens, are also problematic).

- Predetermine and physically impose delimited roads or even railroads for all self-driving cars. If the crux of self-driving cars is to decrease human casualties and increase human productivity, then why not opt for cars in self-driving mode on tracks only and in manual mode on conventional roads?

- Use network connectivity very sparingly, if at all. An Internet of nuclear reactors is an extreme case in point of what the world does not need. Unfortunately, an American nuclear reactor has recently been detected on-line [10, p.17].

In my opinion, self-driving cars should surely *not* depend on any network for their basic operation, even though several specialists advocate highly frequent information exchange for safety's sake.

Admittedly, my analysis has only been partial and, in particular, orthogonal to the topic of deploying self-driving cars in an open environment or in a fully closed one. The latter case entails not only that each car is completely controlled by computers, but also that no pedestrians nor bicyclists can participate in traffic. My recommendation to eschew network-connected cars, especially those that are substantially automated, also holds for fully closed environments.

References

[1] S.M. Casner, E.L. Hutchins, and D. Norman. The Challenges of Partially Automated Driving. *Communications of the ACM*, 59(5):70–77, May 2016.

[2] T.G. Dietterich and E.J. Horvitz. Rise of concerns about AI: Reflections and directions. *Communications of the ACM*, 58(10):38–40, October 2015.

[3] M. Goodman. *Future Crimes: Inside the Digital Underground and the Battle for our Connected World*. Corgi Books, 2016.

[4] M.A. Jackson and E.G. Daylight. *Formalism and Intuition in Software Development*. Lonely Scholar, Geel, August 2015.

[5] N.G. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, January 2012.

[6] D. MacKenzie. *Mechanizing Proof: Computing, Risk, and Trust*. MIT Press, 2004.

[7] K. Mahaffey. Here is how to address car hacking threats, September 2015. https://techcrunch.com/2015/09/12/to-protect-cars-from-cyber-attacks-a-call-for-action/.

[8] P.G. Neumann. Risks of automation: A cautionary total-system perspective of our cyberfuture. *Communications of the ACM*, 59(10):26–30, October 2016.

[9] R.R. Schell. Cyber defense triad for where security matters. *Communications of the ACM*, 59(11):20–23, 2016.

[10] A. Wright. Mapping the internet of things. *Communications of the ACM*, 60(1):16–18, January 2017.