# Antibiotics Time Machines Are Hard to Build

*Ngoc M. Tran and Jed Yang*

*Communicated by Daniel J. Velleman*

ABSTRACT. Could we somehow go back in time before bacteria developed resistance to antibiotics? Mira et al. famously cast antibiotics resistance as an optimization problem called the "antibiotics time machine." We show that such a strategy is NP-hard to compute.

## Antibiotics Time Machines

Resistance to antibiotics in bacteria is a central problem of modern medicine. The work of Mira et al. [MCG⁺15], a group of biologists and mathematicians, cast undoing or avoiding antibiotics resistance as an optimization problem, dubbed the "antibiotics time machine." Their novel approach received wide press coverage. In this article, we define such a time machine mathematically, show that such strategies are NP-hard to compute, and discuss the implications.

Let us think of an antibiotic as a sieve: it kills bacteria with a certain genotype. Repeated applications of the same antibiotic result in a population without this genotype, which renders the antibiotic ineffective. Furthermore, the course of antibiotics has altered the distribution of genotypes, or, in biological terms, the *fitness landscape*, of our bacterial species. This is a critical problem: imagine that the pre-antibiotics bacterial population is held in equilibrium by various factors, such as natural predators, competition for resources, and so on. A new fitness landscape disrupts this equilibrium, potentially leading to a population explosion of these bacteria. One way to mitigate these consequences is to rotate the use of antibiotics that target the same genes but select for different genotypes.

Mira et al. [MCG⁺15] considered the following model. Suppose we have a population of bacteria, all of the same unmutated genotype (called the "wild type"). We are given a set of antibiotics. For each bacterium of a given type, an antibiotic mutates it to another type with some known probability. The problem is to find a sequence of antibiotics (called a treatment plan) that maximizes the fraction of bacteria returning to the wild type. This optimal plan is called an *antibiotic time machine* by Mira et al., alluding to the idea of reversing unnecessary mutations induced in the bacterial population through antibiotic treatments.

Here is a concrete description of the model. Let $S = \{1, 2, \ldots, d\}$ be a set of $d$ states, where each state is a bacterial genotype. A distribution of bacterial population is a vector in the standard $(d-1)$-dimensional simplex $\Delta_{d-1} = \{(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \mid \sum_{i=1}^d x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i\}$, where for $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_d) \in \Delta_{d-1}$, $\mathbf{s}_i$ represents the fraction of the population with genotype $i \in S$. A $d \times d$ matrix is a *transition matrix* if each entry belongs to the interval $[0, 1]$ and each row sums to 1. Let $\mathcal{T}$ be a finite set consisting of $K$ transition matrices, each corresponding to the effect of a given antibiotic on the genotypes of the bacterial population. That is, if $T = (t_{ij})$ is the transition matrix corresponding to some antibiotic, then $t_{ij}$ is the probability that a single bacterium of genotype $i$ will have genotype $j$ after being treated with the antibiotic. As our population consists of millions of bacteria, we can assume that the population evolves deterministically; that is, $t_{ij}$ can be seen as the fraction of bacteria that go from genotype $i$ to $j$ after treatment. Thus, if the population distribution is $\mathbf{s} \in \Delta_{d-1}$ and we apply an antibiotic with transition matrix $T$, then the population distribution after treatment will be $\mathbf{s}T$.

Let $\mathbf{s}, \mathbf{t} \in \Delta_{d-1}$ be the starting and targeted population distributions, respectively. Let $N \in \mathbb{N}$ be the length of the treatment plan. The *time machine* $\mathcal{TM}(d, K, N)$ is the solution to the following optimization problem:

$$\begin{aligned} &\text{maximize} \quad \mathbf{s}T_1T_2 \cdots T_N\mathbf{t}^\top \\ (1) \qquad &\text{subject to} \quad T_1, \ldots, T_N \in \mathcal{T}. \end{aligned}$$

As a concrete example, suppose $\mathbf{s} = \mathbf{t} = (1, 0, \ldots, 0) \in \Delta_{d-1}$. Then $\mathbf{s}T_1T_2 \cdots T_N\mathbf{t}^\top$ is the fraction of bacteria that end up as the wild type after the application of $N$ antibiotics. Our goal is to choose a sequence of antibiotics that will maximize this number.

## Complexity of the Time Machine

How do we solve this problem numerically? Mira et al. [MCG⁺15] used brute force: they enumerated all possible treatment plans. With $K$ antibiotics, there are $K^N$ many treatment plans of length $N$, so this algorithm is not efficient. Mira et al. raised the question of whether an efficient algorithm exists. Unfortunately, we show that the problem is NP-hard; that is, there is likely no polynomial-time algorithm for solving such a problem.

**Theorem 1.** *The time machine optimization problem in* (1) *is NP-hard.*

We first consider an easier decision problem: for a given threshold $\alpha \in (0, 1]$, decide whether there exists a treatment plan of length $N$ such that

$$\begin{aligned} &\mathbf{s}T_1T_2 \cdots T_N\mathbf{t}^\top \geq \alpha \\ (2) \qquad &\text{subject to } T_1, \ldots, T_N \in \mathcal{T}. \end{aligned}$$

Clearly if one can solve (1) in polynomial time, then one can also solve (2) in polynomial time. Our main result states that the latter problem is NP-hard, and thus Theorem 1 follows as a corollary.

**Theorem 2.** *The time machine decision problem in* (2) *is NP-hard.*

*Ngoc Mai Tran is assistant professor of mathematics at the University of Texas at Austin and W-2 professor (Bonn Junior Fellow) at the Hausdorff Center for Mathematics, Germany. Her email address is* ntran@math.utexas.edu.

*Jed Yang is visiting assistant professor of computer science at Carleton College. His email address is* jyang@carleton.edu.

*For permission to reprint this article, please contact:* reprint-permission@ams.org.

DOI: http://dx.doi.org/10.1090/noti1590

Our proof relies on reducing the 3-satisfiability problem (3-SAT) to a special instance of (2) where $\alpha = 1$. Since 3-SAT is NP-hard, we have that (2) is also. Biologically, this means that there are certain inputs where computing an exact solution is likely very time consuming. Furthermore, if $\alpha$ is rational, we may modify our construction to first "discard" a $(1-\alpha)$-fraction of the population, so (2) is hard for infinitely many values of $\alpha$. We provide the details in the appendix.

## Robots and Bacteria

Optimization problems like the time machine also arise in large-scale robotics control. One can equate each bacterium to a robot; each genotype is one of $d$ possible states (such as standing or walking), and each antibiotic is an instruction which changes each robot's state independently at random. The goal is to get all robots to the same state. The difficulty is that one instruction is broadcast to the whole robot population, and thus one cannot optimally control each individual robot. Indeed, if there were only one bacterium instead of a population, then one could apply an antibiotic, check the new state of the bacterium, and repeat. In this case, the time machine is a Markov decision process (MDP) on $d$ states, which can be solved by dynamic programming. When there is a population of bacteria, the population state is a point in the standard $(d-1)$-dimensional simplex $\Delta_{d-1}$, so our MDP has uncountably many states.

In the optimal control literature, our problem is an instance of a *finite-horizon, unobservable MDP with belief-dependent, time-dependent reward function*. This is a controlled stochastic system on finitely many states $S$, with finitely many possible actions $\mathcal{A}$. At discrete time $t = 0, 1, \ldots$, the system is assumed to be in some true but unknown state, and the controller sees only a belief distribution $P_t$ over the set of states $S$. Based on $P_t$, the controller can select an action $a \in \mathcal{A}$, which transits the system to time step $t+1$ with belief distribution $P_{t+1} = f(P_t, a)$ and gives reward $r_t(P_{t+1})$. A *policy* is a sequence of actions. In finite-horizon problems, the goal is to choose a policy that maximizes the total expected reward after $T$ time steps. In the antibiotics context, a state is a genotype, $P_t$ is the distribution of genotypes in the bacterial population at time $t$, and an action is an antibiotic. The reward function $r_t$ is identically zero for $t < T$, while $r_T(P_{T+1}) = (P_{T+1})_1$. Finite-horizon unobservable MDPs are known to be NP-hard [LGM01], with a reduction to $k$-SAT similar to our proof of Theorem 2.

## How to Build a Time Machine Anyway

Exact algorithms for unobservable MDPs use incremental pruning, and this idea also yields an exact algorithm for the antibiotics time machine. The premise is to enumerate by brute force but eliminate suboptimal treatment plans early. To illustrate, suppose we are able to solve the antibiotics time machine with $N = 1$ for each possible starting state $\mathbf{s} \in \Delta_{d-1}$. Let $T$ be an antibiotic that does not achieve the maximum in (1) for any $\mathbf{s}$. Then for the antibiotics time machine with any $N \geq 1$ and any starting point, the optimal treatment plan cannot end with $T$. If there are $K'$ such suboptimal antibiotics, then the total number of treatment plans to consider is now $K^{T-1}(K-K')$, as opposed to $K^T$. Repeated application of this argument for $N = 2, 3, \ldots$ amounts to enumerating the treatment plans in backward time steps while discarding suboptimal branches early.

One can also save computation time by restricting or approximating the belief space. Let us again consider the time machine in one step. As a function of the starting state $\mathbf{s}$, the objective function in (1) is the maximum of $K$ linear functions, so it is convex and piecewise linear. Its graph subdivides $\Delta_{d-1}$ into at most $K$ full-dimensional polytopes, where all starting points $\mathbf{s}$ in the same polytope are optimized by the same antibiotic unique to that polytope. Now, suppose we know that $\mathbf{s}$ can only lie in some strict subset $\Delta'$ of $\Delta_{d-1}$. If $T$ is an antibiotic whose polytope does not intersect $\Delta'$, then $T$ is also suboptimal. Combined with pruning, this reduces the set of possible optimal treatment plans even further.

## Practical Considerations: The Efficient, Approximate Time Machine?

Real-life instances of the time machine make exact computations prohibitive. To put the problem scale into perspective, the particular time machine of Mira et al. [MCG+15] had 16 genotypes and 15 antibiotics and used the brute force algorithm up to $N = 6$, that is, over $15^6 \approx 1.1 \times 10^7$ treatment plans. In comparison, the methicillin-resistant *Staphylococcus aureus* (MRSA), a "superbug" common in healthcare centers, has about 30 genotypes identified [SHIB+06]. Currently there are six antibiotics recommended for treatment of MRSA. A standard therapy is 28 days, with antibiotics applied between one and three times a day. Altogether this gives a time machine problem with $d \approx 30$, $K \approx 5$ to 10, and $N \approx 30$ to 90. The smallest instance with $(d, K, N) = (30, 5, 30)$ already creates $5^{30} \approx 9.3 \times 10^{20}$ possible treatment plans. Unless the transition matrices have very special structures, such instances cannot be solved exactly.

*another significant contribution of mathematics to society*

This is where connections to robotics could be very valuable. Unobservable MDPs are special instances of partially observable MDPs (POMDPs), which find extensive applications in robotics and motion planning [DS13]. Large-scale approximation algorithms for these problems use a combination of belief space approximation and early pruning of suboptimal policies, as discussed above. However, their reward functions are not time or belief dependent. If such approximation algorithms can be adapted to the antibiotics time machine, one could solve real-life instances with tens of thousands of states and antibiotics. That would be yet another significant contribution of mathematics to society, one that even has a cool name.

## Appendix A. Reduction Construction and Proof of the Main Theorem

Let $X = \{x_1, \dots, x_n\}$ be a set of Boolean variables taking Boolean values $\{+, -\}$. A triple $(\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k)$ with $\varepsilon_i, \varepsilon_j, \varepsilon_k \in \{+, -\}$ is called a *clause*. Let $C = \{c_1, \dots, c_m\}$ be a set of clauses. Write $(x_i = \nu_i)_i$ to mean $x_i = \nu_i$ for $i = 1, \dots, n$. An *assignment* $(x_i = \nu_i)_i$ *satisfies* a clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k)$ if $(\varepsilon_i, \varepsilon_j, \varepsilon_k) \neq (\nu_i, \nu_j, \nu_k)$. The 3-SAT problem is: given a set of variables and clauses, decide whether there exists an assignment that satisfies all clauses. We will assume that each variable occurs (negated or not) in at least one clause. In this section, we reduce a 3-SAT instance to an instance of (2) by constructing an appropriate time machine.

### Construction of the Time Machine

Let $N = m + 2$ and $\alpha = 1$. Consider a time machine with $d = 3n + m + 3$ states and $K = 7m + 2$ transition matrices. Index these $d$ states as follows: For each of the $n$ variables $x \in X$, there are states $\mathbf{x}$, $\mathbf{x}^-$, and $\mathbf{x}^+$. For each of the $m$ clauses $c \in C$, there is a state $\mathbf{c}$. Finally, the last three states are the start state $\mathbf{s}$, the "death" state $\mathbf{d}$, and the "tally" state $\mathbf{f}$. We identify each state with its characteristic (row) vector and define each transition matrix by its action on these basis vectors.

If $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in C$, then for each choice of $(\nu_i, \nu_j, \nu_k) \in \{+, -\}^3 \setminus \{(\varepsilon_i, \varepsilon_j, \varepsilon_k)\}$, define a transition matrix $T = T_c^{\nu_i, \nu_j, \nu_k}$ as follows:

$$\mathbf{z}T = \begin{cases} \mathbf{d}, & \mathbf{z} = \mathbf{s}, \\ \mathbf{f}, & \mathbf{z} = \mathbf{c}, \\ \mathbf{x}_\ell^{\nu_\ell}, & \mathbf{z} = \mathbf{x}_\ell,\ \ell \in \{i, j, k\}, \\ \mathbf{d}, & \mathbf{z} = \mathbf{x}_\ell^{-\nu_\ell},\ \ell \in \{i, j, k\}, \\ \mathbf{z}, & \text{otherwise.} \end{cases}$$

Define a starting matrix $S$ by

$$\mathbf{z}S = \begin{cases} p\left(\sum_{x \in X} \mathbf{x} + \sum_{c \in C} \mathbf{c}\right), & \mathbf{z} = \mathbf{s}, \\ \mathbf{d}, & \text{otherwise,} \end{cases}$$

where $p = 1/(n + m)$. Lastly, define a final matrix $F$ by

$$\mathbf{z}F = \begin{cases} \mathbf{s}, & \mathbf{z} \in \{\mathbf{x}_i^{\nu_i} : \nu_i \in \{+, -\}\}, \\ \mathbf{s}, & \mathbf{f}, \\ \mathbf{d}, & \text{otherwise.} \end{cases}$$

Let $\mathcal{T}$ consist of $S$, $F$, and the $7m$ transition matrices $T_c^{\nu_i, \nu_j, \nu_k}$ defined above. This concludes the reduction construction.

### Proof of Reduction

We shall prove that with the above time machine constructed from the 3-SAT problem, (2) solves the 3-SAT problem with the given clauses. Since the latter is NP-hard, this implies that (2) is also. We refer to the fraction of the bacterial population in a given state as its *weight*. First, suppose $(x_i = \nu_i)_i$ is a satisfying assignment. Apply $S$ to $\mathbf{s}$. For each $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in C$, apply $T_c^{\nu_i, \nu_j, \nu_k}$, which exists as $c$ is satisfied. Finally, apply $F$.

**Lemma 3.** *The sequence of matrices given above sends all weight back to* $\mathbf{s}$. *That is, it is a solution of* (2) *for* $\alpha = 1$.

*Proof.* For each $x \in X$, the matrix $S$ sends some weight to $\mathbf{x}$. The weight is moved to $\mathbf{x}^{\nu_i}$ the first time $\pm x$ occurs in a clause $c$ and is moved back to $\mathbf{s}$ by $F$ at the end. For each $c \in C$, the matrix $S$ sends some weight to $\mathbf{c}$. This weight is moved to $\mathbf{f}$ by $T_c^{\nu_i, \nu_j, \nu_k}$ and is moved back to $\mathbf{s}$ by $F$ at the end. Therefore all the weights return to $\mathbf{s}$, as desired. $\square$

Conversely, suppose there is a sequence $T_1, T_2, \dots, T_N$ of transition matrices so that

$$\mathbf{s}T_1 T_2 \cdots T_N \mathbf{t}^\top = 1.$$

The aim is to extract a satisfying assignment.

Consider the process of applying the transition matrices $T_i$ to $\mathbf{s}$ sequentially in $N$ steps. Note that any weight at $\mathbf{d}$ stays at $\mathbf{d}$ forever. As such, to achieve full weight at $\mathbf{s}$ after $N$ steps, the state $\mathbf{d}$ cannot receive weight at any point in the process.[1] Consequently, it is clear that the first matrix to apply has to be $S$, as we have $T(\mathbf{s}) = \mathbf{d}$ for $T \in \mathcal{T} \setminus \{S\}$.

The only way for $\mathbf{s}$ to gain weight is to apply $F$. Prior to applying $F$ for the first time, all weights must be supported on the $\mathbf{x}_i^\pm$ and $\mathbf{f}$, so as to avoid losing any weight to the death state $\mathbf{d}$. In particular, for each clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in C$, state $\mathbf{c}$ must no longer carry weight at this point. That is, after $\mathbf{c}$ receives some weight by $S$ in the first step, it must subsequently lose the weight, which can only be achieved by applying an associated matrix $T_c^{\nu_i, \nu_j, \nu_k}$ for some choice of $(\nu_i, \nu_j, \nu_k)$. This takes (at least) one step for each clause. Since the sequence of matrices is of length precisely $N = m + 2$, we conclude that the sequence starts with $S$, finishes with $F$, and contains precisely one matrix corresponding to each clause.

When $S$ is applied, the full weight at $\mathbf{s}$ is split into $n + m$ *packets*, each of weight $p = 1/(n + m)$. Since no other matrices split weights, we may consider the remaining process as a discrete system moving each packet as a unit. We already saw that the $m$ packets associated to the clauses are moved to $\mathbf{f}$ during the middle $m$ steps. It remains to analyze the remaining $n$ packets associated to the variables. To avoid moving any weight to the death state $\mathbf{d}$, the packet at $\mathbf{x}_i$ can only be moved to $\mathbf{x}_i^+$ or $\mathbf{x}_i^-$. Once this happens, the packet can only be moved again by $F$ at the last step. So at the penultimate step, there is a packet on $\mathbf{x}_i^{\tilde{\nu}_i}$ for exactly one choice of $\tilde{\nu}_i$ for each $i$. The following lemma finishes the proof.

**Lemma 4.** *For each $i$, let $\tilde{\nu}_i$ be such that $\mathbf{x}_i^{\tilde{\nu}_i}$ has nonzero weight after $m + 1$ steps, i.e.,*

$$\mathbf{s}T_1 T_2 \cdots T_{N-1} (\mathbf{x}_i^{\tilde{\nu}_i})^\top > 0.$$

*Then $(x_i = \tilde{\nu}_i)_i$ is a satisfying assignment.*

[1] *Intuitively,* $\mathbf{d}$ *is a "death" state, where bacteria go to die, never to be recovered to the wild type.*

Indeed, consider a clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in C$. We know that (exactly) one associated transition matrix $T_c^{\nu_i,\nu_j,\nu_k}$ is used. Suppose, towards a contradiction, that $\tilde{\nu}_\ell \neq \nu_\ell$ for some $\ell \in \{i, j, k\}$. After applying $T_c^{\nu_i,\nu_j,\nu_k}$, the packet corresponding to $x_\ell$ is at $\mathbf{x}_\ell^{\nu_\ell}$ or $\mathbf{d}$, with no way of moving to $\mathbf{x}_\ell^{\tilde{\nu}_\ell}$, a contradiction. So $(\tilde{\nu}_i, \tilde{\nu}_j, \tilde{\nu}_k) = (\nu_i, \nu_j, \nu_k) \neq (\varepsilon_i, \varepsilon_j, \varepsilon_k)$ by construction, implying that clause $c$ is satisfied. □

This shows that a 3-SAT instance has a solution if and only if the associated time machine can attain a threshold of 1. We therefore conclude that the time machine decision problem is NP-hard, as desired.

## Acknowledgments

The authors wish to thank Bernd Sturmfels and Joel Kileel for stimulating discussions. The first-named author is supported by an award from the Simons Foundation (#197982 to The University of Texas at Austin). The second-named author is supported by NSF RTG grant NSF/DMS-1148634.

## References

[DS13] Alain Dutech and Bruno Scherrer, Partially observable markov decision processes, *Markov Decision Processes in Artificial Intelligence*, ISTE, London, 2010, pp. 187–227. MR 2808773

[LGM01] Christopher Lusena, Judy Goldsmith, and Martin Mundhenk, Nonapproximability results for partially observable Markov decision processes, *J. Artificial Intelligence Res.* **14** (2001), no. 1, 83–113. MR 1835269

[MCG$^+$15] Portia M. Mira, Kristina Crona, Devin Greene, Juan C. Meza, Bernd Sturmfels, and Miriam Barlow, Rational design of antibiotic treatment plans: A treatment strategy for managing evolution and reversing resistance, *PLOS ONE* **10** (2015), no. 5, e0122283.

[SHIB$^+$06] Alex J. Stephens, Flavia Huygens, John Inman-Bamber, Erin P. Price, Graeme R. Nimmo, Jacqueline Schooneveldt, Wendy Munckhof, and Philip M. Giffard, Methicillin-resistant *Staphylococcus aureus* genotyping using a small set of polymorphisms, *J. of Medical Microbiology* **55** (2006), no. 1, 43–51.

## Photo Credits

Opening photo courtesy of Getty Images.
Photo of Ngoc Mai Tran courtesy of Barbara Frommann.
Photo of Jed Yang courtesy of Jed Yang.

## ABOUT THE AUTHORS

**Ngoc Mai Tran** loves to bring new applications to tropical geometry and probability. She also likes manga, strategy games, and traveling.

Ngoc Mai Tran

**Jed Yang** enjoys applying computational complexity techniques to questions from various disciplines. In his spare time, he likes to think about mathematics.

Jed Yang