Antibiotics Time Machine is NP-hard

NGOC M TRAN AND JED YANG

Abstract. The antibiotics time machine is an optimization question posed by Mira *et al.* on the design of antibiotic treatment plans to minimize antibiotic resistance. The problem is a variation of the Markov decision process. These authors asked if the problem can be solved efficiently. In this paper, we show that this problem is NP-hard in general.

1. Problem statement and results

Resistance to antibiotics in bacteria is a central problem in medicine. One method for mitigating its consequences is to rotate the use of antibiotics. A natural question arises: what is the optimal sequence of antibiotics one should apply? In this work, we follow the model proposed by Mira *et al.* [MCG+14]. 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 bacteria 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 is equivalent to minimizing the fraction of antibiotics-induced mutations in the bacterial population.

Here is a concrete description of the model. Let S be a set of d states, where each state is a bacterial genotype. A $d \times d$ matrix $T = (t_{ij})$ is a **transition matrix** if $t_{ij} \in [0, 1]$ 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, t_{ij} is the probability that a bacteria of genotype i will have genotype j after being treated with the corresponding antibiotic. The standard $(d-1)$-dimensional simplex Δ^{d-1} is given by $\Delta^{d-1} = \{(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \mid \sum_{i=1}^{d} x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i\}$. Let $\mathbf{s} = (1, 0, \ldots, 0) \in \Delta^{d-1}$ be the starting state of the population, where all bacteria are of the first state (the wild type). 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

$$\begin{aligned} \text{maximize} \quad & \mathbf{s} T_1 T_2 \cdots T_N \mathbf{s}^\top \\ \text{subject to} \quad & T_1, \ldots, T_N \in \mathcal{T}. \end{aligned} \tag{1}$$

The term *time machine*, coined by Mira *et al.*, alludes to the idea of reversal to the wild type of unnecessary mutations done by antibiotic treatments. They proposed and solved a specific numerical instance of the time machine for 16 genotypes of the TEM β-lactamase, with a specific model of mutation, and a set of up to six antibiotics. Their algorithm is to enumerate all possible treatment plans of a given length with the given set of antibiotics and output the optimal one.

With K antibiotics, there are K^N many treatment plans of length N, and thus direct enumeration is not an efficient algorithm. Mira *et al.* raised the question of whether an efficient algorithm exists. In this paper, we show that the general problem is NP-hard.

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

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

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

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.*

Our proof relies on reducing 3-SAT to a special instance of (2) where $\alpha = 1$. Since 3-SAT is NP-hard [Kar72], we have that (2) is also. The reduction construction is described in Section 2. The proof that this is a correct reduction is in Section 3.

1.1. **Related literature.** The time machine may also arise in large scale robotics control, where one has a large number of robots with d possible states, and each T_i is an instruction which changes each robot's state independently at random. In general, it can be viewed as a Markov decision problem with infinitely many states. Indeed, if there was only one bacteria instead of a population, one may apply an antibiotic T_i, check the new state of the bacteria, and repeat. In this case, the time machine is a Markov decision process 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 Δ^{d-1}, which is an infinite set.

There is a large literature on Markov decision processes, see [FSA02] and references therein. However, we are unaware of any existing results on the time machine, or the equivalent of Theorem 1. Closely related to the time machine is the partially observable Markov decision process [DS13], where the underlying system is assumed to be in one of the unobservable d states, and $s \in \Delta^d$ reflects the uncertainty over the true state. It would be interesting to see if techniques from this setup can yield polynomial time approximations to the time machine.

1.2. **Open problems.** Many interesting questions remain on the antibiotics time machine. We mention two examples. The last question was raised by Joe Kileel.

(i) For what classes of antibiotics $\mathcal{T}$ is the problem solvable in polynomial time? Are there biologically relevant instances?

(ii) When does the limit $\lim_{N\to\infty} \max_{T_i\in\mathcal{T}} \mathbf{s}T_1T_2\cdots T_N\mathbf{s}^\top$ exist? If so, how fast is the convergence? In this case, for large N, can one produce an approximate time machine in polynomial time?

2. Reduction construction

Let $X = \{x_1,\ldots,x_n\}$ 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 $\mathcal{C} = \{c_1,\ldots,c_m\}$ be a set of clauses. Write $(x_i = v_i)_i$ to mean $x_i = v_i$ for $i = 1,\ldots,n$. An **assignment** $(x_i = v_i)_i$ **satisfies** a clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k)$ if $(\varepsilon_i,\varepsilon_j,\varepsilon_k) \neq (v_i,v_j,v_k)$. Without loss of generality, assume each $x \in X$ actually occurs (negated or not) somewhere. The 3-SAT problem is: given a set of variables and clauses, decide whether there exists an assignment that satisfies all clauses. In this section, we reduce a 3-SAT instance to an instance of (2).

Let $N = m + 2$ and $\alpha = 1$. Create $d = 3n + m + 3$ states and $K = 7m + 2$ transition matrices. Create the start state $\mathbf{s}$, a "death" state $\mathbf{d}$, and a "tally" state $\mathbf{f}$. For each variable $x \in X$, create states $\mathbf{x}, \mathbf{x}^-, \mathbf{x}^+$. For each clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in \mathcal{C}$, create state $\mathbf{c}$. We identify each state with its characteristic (row) vector and define each transition matrix by its action on these basis vectors and extend linearly.

For each choice of $(v_i, v_j, v_k) \in \{+, -\}^3 \smallsetminus \{(\varepsilon_i, \varepsilon_j, \varepsilon_k)\}$, create a transition matrix $T = T_c^{v_i, v_j, v_k}$ as follows:

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

Define a starting matrix S by

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

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

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

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

3. Proof of Main Theorem

We shall prove that with the setup in Section 2, the time machine decision 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 $(x_i = v_i)_i$ is a satisfying assignment. Apply S to $\mathbf{s}$. For each $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in \mathcal{C}$, apply $T_c^{v_i, v_j, v_k}$, which exists as c is satisfied. Finally, apply F.

Lemma 3. *The sequence of matrices given above sends all weights 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 $\mathbf{x}^{v_i}$ the first time $\pm x$ occurs in a clause c, and is moved back to $\mathbf{s}$ by F at the end. For each $c \in \mathcal{C}$, the matrix S sends some weight to $\mathbf{c}$. This weight is moved to $\mathbf{f}$ by $T_c^{v_i, v_j, v_k}$, and is moved back to $\mathbf{s}$ by F at the end. Therefore all the weights returns to $\mathbf{s}$, as desired. □

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

$$\mathbf{s}T_1T_2 \cdots T_N\mathbf{s}^\top = 1.$$

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 $T \in \mathcal{T} \smallsetminus \{S\}$.

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 $\mathbf{f}$, as to avoid losing any weight to the death state $\mathbf{d}$. In particular, for each clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in \mathcal{C}$, 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 $T_c^{v_i,v_j,v_k}$ for some choice of (v_i, v_j, v_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 $\mathbf{x}_i^{\widetilde{v}_i}$ for exactly a choice of $\widetilde{v}_i$ for each i. The following lemma finishes the proof.

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

$$\mathbf{s}T_1T_2 \cdots T_{N-1}(\mathbf{x}_i^{\widetilde{v}_i})^\top > 0.$$

Then $(x_i = \widetilde{v}_i)_i$ is a satisfying assignment.

Indeed, consider a clause $c = (\varepsilon_i x_i, \varepsilon_j x_j, \varepsilon_k x_k) \in \mathcal{C}$. We know that (exactly) one associated transition matrix $T_c^{v_i,v_j,v_k}$ is used. Suppose, towards a contradiction, that $\widetilde{v}_\ell \neq v_\ell$ for some $\ell \in \{i, j, k\}$. After applying $T_c^{v_i,v_j,v_k}$, the packet corresponding to x_ℓ is at $\mathbf{x}_\ell^{v_\ell}$ or $\mathbf{d}$, with no way of moving to $\mathbf{x}_\ell^{\widetilde{v}_\ell}$, a contradiction. So $(\widetilde{v}_i, \widetilde{v}_j, \widetilde{v}_k) = (v_i, v_j, v_k) \neq (\varepsilon_i, \varepsilon_j, \varepsilon_k)$ 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.

Acknowledgements. 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*, pages 185–228, 2013.

[FSA02] Eugene A Feinberg, Adam Shwartz, and Eitan Altman. *Handbook of Markov decision processes: methods and applications*. Kluwer Academic Publishers Boston, MA, 2002.

[Kar72] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[MCG+14] Portia M Mira, Kristina Crona, Devin Greene, Juan C Meza, Bernd Sturmfels, and Miriam Barlow. Rational design of antibiotic treatment plans. *arXiv preprint arXiv:1406.1564*, 2014.

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

Department of Mathematics, The University of Texas at Austin, TX 78751, USA
E-mail address: ntran@math.utexas.edu

School of Mathematics, The University of Minnesota, MN 55455, USA
E-mail address: jedyang@umn.edu