# *Programming* Biological Time Machines: Lecture I[1]

Burak Kocuk

Sabancı University

17 November 2023

[1]Based on: Kocuk, B. (2022). Optimization problems involving matrix multiplication with applications in materials science and biology. Engineering Optimization, 54(5), 786-804.

# Disclaimer

- I am, by no means, a "biology guy". In fact, biology was my least favorite topic in high school!
- I will explain the problem the way I understand it, that is, from an operations research/optimization point of view.
- I concur that the mathematics in this presentation is not very deep but it combines a few tricks nicely to solve an interesting problem.

# Outline

# Outline

# A "Simple" Optimization Problem Example

- You would like to pack your *knapsack* for the vacation in such a way that the total value of the items that you put is maximized.

# A "Simple" Optimization Problem Example

- You would like to pack your *knapsack* for the vacation in such a way that the total value of the items that you put is maximized.
- Here is an *instance* of the problem:
  - Suppose we have 4 items to choose from.
  - The value of these items are 5, 7, 4, 3.
  - The size of these items are 8, 11, 6, 4.
  - The capacity of the knapsack 14.

# A "Simple" Optimization Problem Example

- You would like to pack your *knapsack* for the vacation in such a way that the total value of the items that you put is maximized.
- Here is an *instance* of the problem:
  - Suppose we have 4 items to choose from.
  - The value of these items are 5, 7, 4, 3.
  - The size of these items are 8, 11, 6, 4.
  - The capacity of the knapsack 14.
- Can you give an upper bound on the number of ways you can organize your knapsack?

# A "Simple" Optimization Problem Example

- You would like to pack your *knapsack* for the vacation in such a way that the total value of the items that you put is maximized.
- Here is an *instance* of the problem:
  - Suppose we have 4 items to choose from.
  - The value of these items are 5, 7, 4, 3.
  - The size of these items are 8, 11, 6, 4.
  - The capacity of the knapsack 14.
- Can you give an upper bound on the number of ways you can organize your knapsack?
- What are the best items to select?

# A "Simple" Optimization Problem Example

- You would like to pack your *knapsack* for the vacation in such a way that the total value of the items that you put is maximized.
- Here is an *instance* of the problem:
  - Suppose we have 4 items to choose from.
  - The value of these items are 5, 7, 4, 3.
  - The size of these items are 8, 11, 6, 4.
  - The capacity of the knapsack 14.
- Can you give an upper bound on the number of ways you can organize your knapsack?
- What are the best items to select?
- Do you think this is an *easy* problem or a *hard* problem to solve if we have *many* items?

# A Motivating Example: Knapsack Problem

- Here is some notation (*parameters* of the problem):
  - $I$: the set of items
  - $v_i$: value of item $i \in I$
  - $s_i$: size of item $i \in I$
  - $c$: capacity of the knapsack

# A Motivating Example: Knapsack Problem

- Here is some notation (*parameters* of the problem):
  - $I$: the set of items
  - $v_i$: value of item $i \in I$
  - $s_i$: size of item $i \in I$
  - $c$: capacity of the knapsack
- Here is the optimization approach: Let us define a *decision variable* $x_i$ taking value 1 if item $i$ is selected and 0 otherwise.

# A Motivating Example: Knapsack Problem

- Here is some notation (*parameters* of the problem):
  - $I$: the set of items
  - $v_i$: value of item $i \in I$
  - $s_i$: size of item $i \in I$
  - $c$: capacity of the knapsack
- Here is the optimization approach: Let us define a *decision variable* $x_i$ taking value 1 if item $i$ is selected and 0 otherwise.
- We want to solve the problem

$$\max_x \sum_{i \in I} v_i x_i \qquad \text{(objective function)}$$

$$\text{s.t.} \sum_{i \in I} s_i x_i \leq c \qquad \text{(constraint)}$$

$$x_i \in \{0, 1\} \qquad i \in I. \qquad \text{(decision variable domain)}$$

# A Motivating Example: Knapsack Problem

- Here is some notation (*parameters* of the problem):
  - $I$: the set of items
  - $v_i$: value of item $i \in I$
  - $s_i$: size of item $i \in I$
  - $c$: capacity of the knapsack
- Here is the optimization approach: Let us define a *decision variable* $x_i$ taking value 1 if item $i$ is selected and 0 otherwise.
- We want to solve the problem

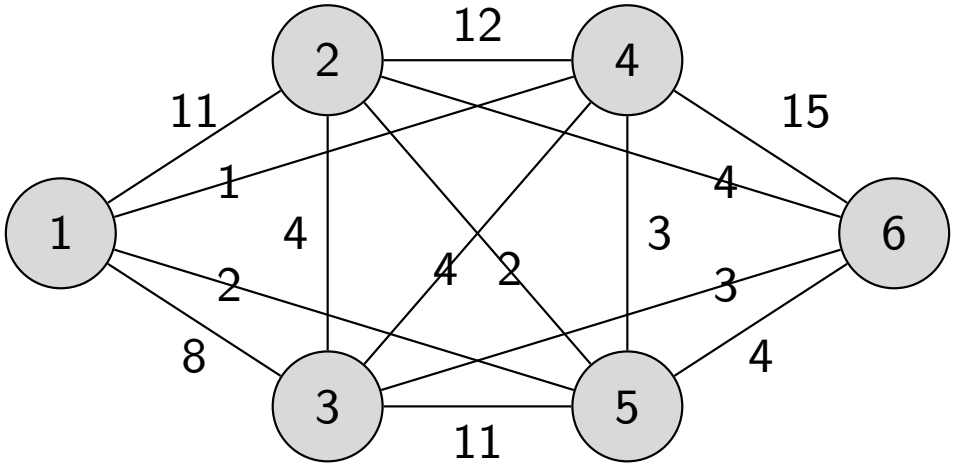
$$\max_x \sum_{i \in I} v_i x_i \qquad \text{(objective function)}$$

$$\text{s.t.} \sum_{i \in I} s_i x_i \leq c \qquad \text{(constraint)}$$

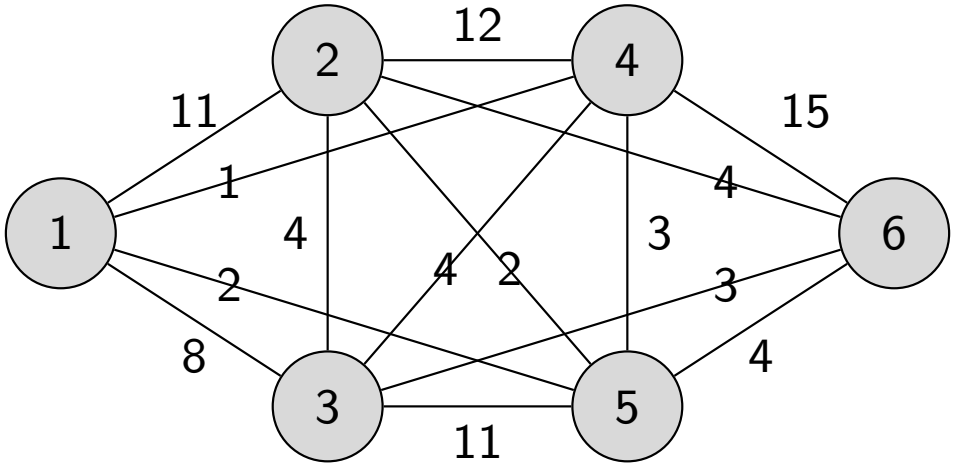
$$x_i \in \{0, 1\} \qquad i \in I. \qquad \text{(decision variable domain)}$$

- There are efficient and general-purpose solvers for *this kind* of problems!

# Two More Motivating Examples

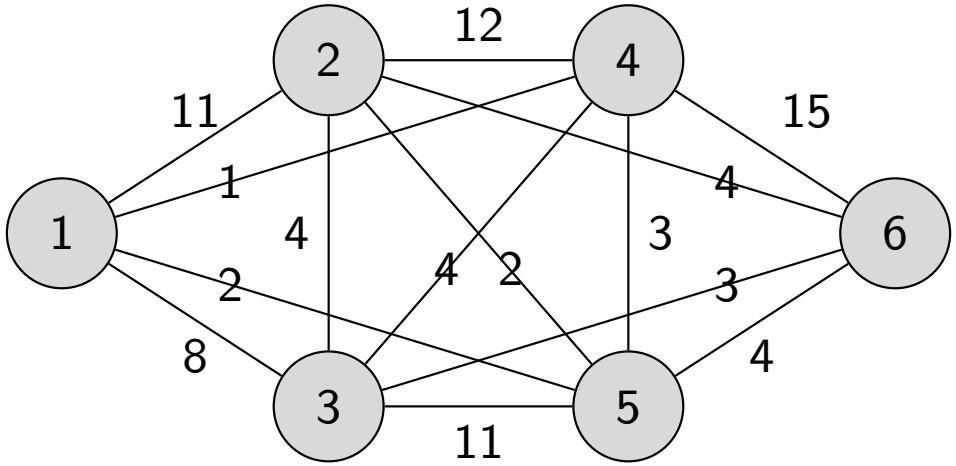


# Two More Motivating Examples




- Shortest Path Problem: Given a network, a source and a sink, find the shortest path.
  - Easy in theory (polynomially solvable) and also in practice.

# Two More Motivating Examples




- Shortest Path Problem: Given a network, a source and a sink, find the shortest path.
  - Easy in theory (polynomially solvable) and also in practice.
- Traveling Salesperson Problem: Given a network, find the shortest tour that visits all the nodes exactly once.
  - Difficult in theory (NP-Hard) but large instances (tens of thousand of nodes) can be solved optimally.

# What is Optimization/Mathematical Programming?

George B. Dantzig: "If the system exhibits a structure which can be represented by a mathematical equivalent, called a mathematical model, and if the objective can be also so quantified, then some computational method may be evolved for choosing the best schedule of actions among alternatives. Such use of mathematical models is termed mathematical programming."

# What is Optimization/Mathematical Programming?

George B. Dantzig: "If the system exhibits a structure which can be represented by a mathematical equivalent, called a mathematical model, and if the objective can be also so quantified, then some computational method may be evolved for choosing the best schedule of actions among alternatives. Such use of mathematical models is termed mathematical programming."

A generic optimization problem looks like this:

$$\begin{aligned} \min_x \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

Here, the functions $f$, $g_i$, $h_l$ are real-valued and well-defined.

# Solving an Optimization Problem

- Consider the problem

$$\begin{aligned}
&\min_{x} \ f(x) && && \text{(objective function)}\\
&\text{s.t. } g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)}\\
&\quad\ h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)}\\
&\quad\ x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)}
\end{aligned}$$

# Solving an Optimization Problem

- Consider the problem

$$\begin{aligned} \min_{x} \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

- Solving an optimization problem means finding an optimal solution (and proving its optimality).

# Solving an Optimization Problem

- Consider the problem

$$\begin{aligned} \min_{x} \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

- Solving an optimization problem means finding an optimal solution (and proving its optimality).
- An engineer might be satisfied with a feasible solution that "looks good".
  - Domain knowledge, local solvers, discretization, heuristics,...

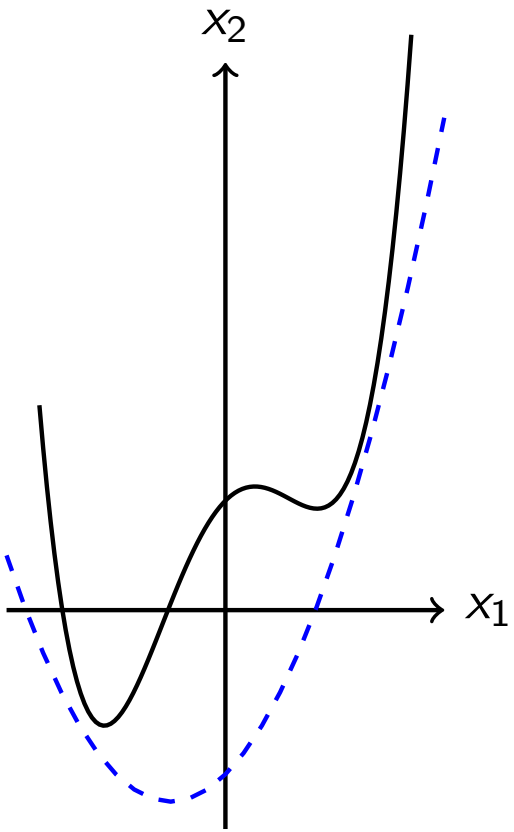
# Solving an Optimization Problem

- Consider the problem

$$\begin{aligned} \min_x \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

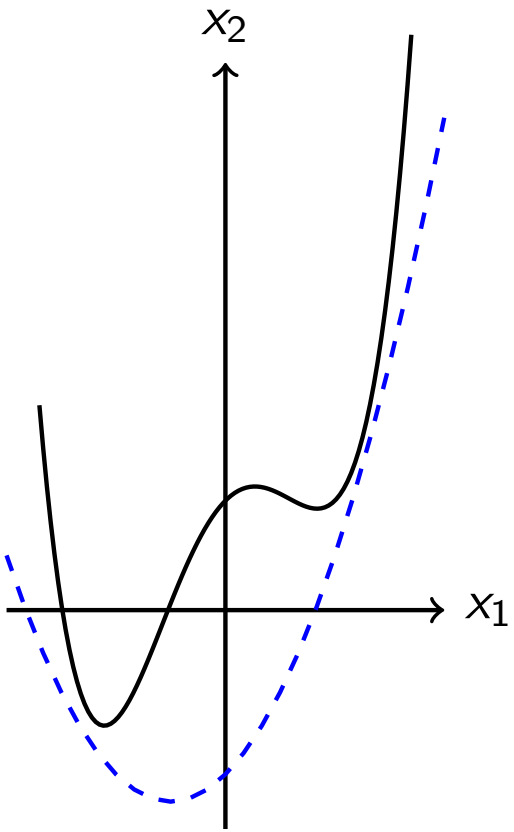
- Solving an optimization problem means finding an optimal solution (and proving its optimality).
- An engineer might be satisfied with a feasible solution that "looks good".
  - Domain knowledge, local solvers, discretization, heuristics,...
- An optimizer would not be completely satisfied without a certificate of (near-) optimality.
  - Relaxation, convexification, (spatial) branch-and-bound, cutting planes,...

# The Concept of Relaxation



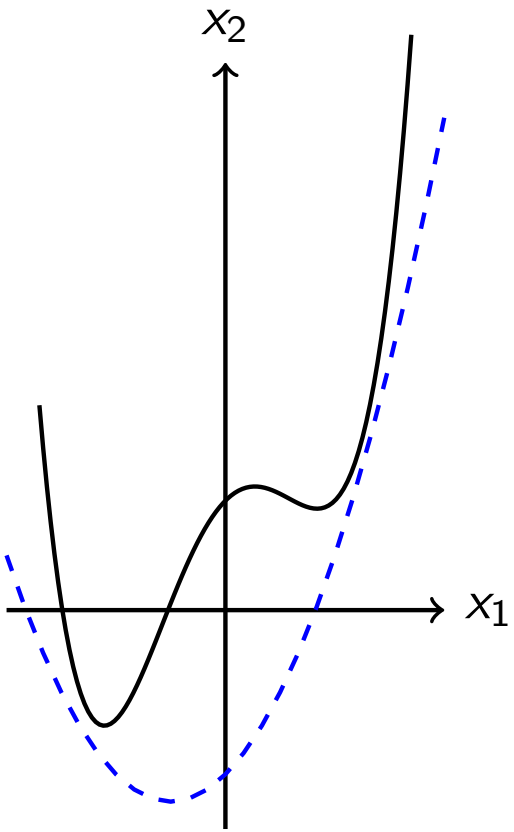

- Suppose that we want to minimize the nonconvex black function.
  - Any feasible solution in the domain gives an upper bound.

# The Concept of Relaxation



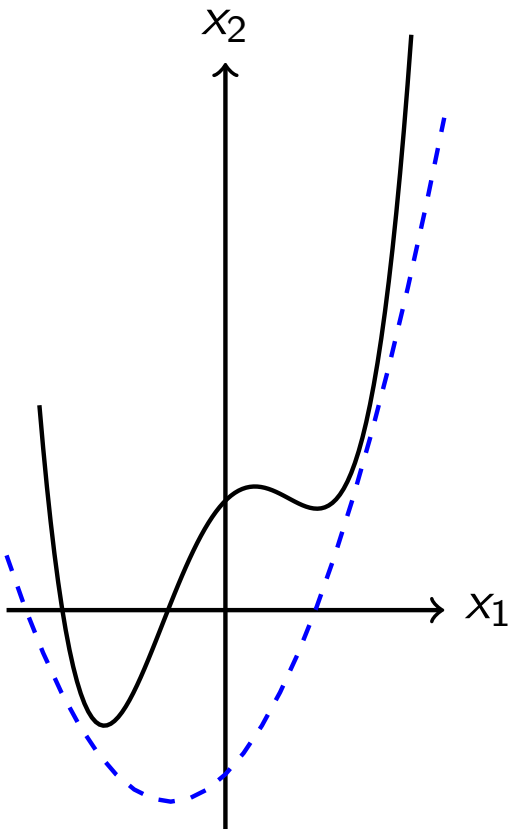

- Suppose that we want to minimize the nonconvex black function.
  - Any feasible solution in the domain gives an upper bound.
- Suppose that we also minimize the convex blue underestimator.
  - The minimum of the underestimator gives a lower bound.

# The Concept of Relaxation




- Suppose that we want to minimize the nonconvex black function.
  - Any feasible solution in the domain gives an upper bound.
- Suppose that we also minimize the convex blue underestimator.
  - The minimum of the underestimator gives a lower bound.
- We can divide the domain into subdomains and obtain a global optimization algorithm.

# The Concept of Relaxation




- Suppose that we want to minimize the nonconvex black function.
  - Any feasible solution in the domain gives an upper bound.
- Suppose that we also minimize the convex blue underestimator.
  - The minimum of the underestimator gives a lower bound.
- We can divide the domain into subdomains and obtain a global optimization algorithm.
- Relaxations allow us to prove (near-) optimality WITHOUT exploring all solutions!

# When is an Optimizer Happy/Hopeful?

- Recall the problem

$$\begin{aligned} \min_{x} \; & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

# When is an Optimizer Happy/Hopeful?

- Recall the problem

$$\begin{aligned} \min_x \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

- Linear Programming: All functions linear, no integer variables.
  - An optimizer is usually very happy in this case.

# When is an Optimizer Happy/Hopeful?

- Recall the problem

$$\begin{aligned} \min_x \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

- Linear Programming: All functions linear, no integer variables.
  - An optimizer is usually very happy in this case.
- Mixed-Integer Linear Programming (MILP): All functions linear.
  - An optimizer is usually happy in this case.

# When is an Optimizer Happy/Hopeful?

- Recall the problem

$$\begin{aligned} \min_x \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

- Linear Programming: All functions linear, no integer variables.
  - An optimizer is usually very happy in this case.
- Mixed-Integer Linear Programming (MILP): All functions linear.
  - An optimizer is usually happy in this case.
- Convex (Conic) Programming: All functions convex (conic representable), no integer variables.
  - An optimizer is usually happy in this case.

# When is an Optimizer Happy/Hopeful?

- Recall the problem

$$\begin{aligned} \min_x \ & f(x) && && \text{(objective function)} \\ \text{s.t. } & g_i(x) \leq 0 && i = 1, \ldots, m && \text{(inequality constraints)} \\ & h_l(x) = 0 && l = 1, \ldots, k && \text{(equality constraints)} \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}. && && \text{(decision variable domain)} \end{aligned}$$

- Linear Programming: All functions linear, no integer variables.
  - An optimizer is usually very happy in this case.
- Mixed-Integer Linear Programming (MILP): All functions linear.
  - An optimizer is usually happy in this case.
- Convex (Conic) Programming: All functions convex (conic representable), no integer variables.
  - An optimizer is usually happy in this case.
- Mixed-Integer Convex (Conic) Programming: All functions convex (conic representable).
  - An optimizer is usually hopeful in this case.

# Progress in General Purpose MILP Solvers

- MILP solvers are more than $10^6$ faster than they were 30 years ago (hardware improvement excluded).

# Progress in General Purpose MILP Solvers

- MILP solvers are more than $10^6$ faster than they were 30 years ago (hardware improvement excluded).
- This enormous progress is due to the theoretical advances in understanding problem structure better, which led to more efficient algorithms.

# Progress in General Purpose MILP Solvers

- MILP solvers are more than $10^6$ faster than they were 30 years ago (hardware improvement excluded).
- This enormous progress is due to the theoretical advances in understanding problem structure better, which led to more efficient algorithms.
- Relevance to Time Machines: The static-deterministic version can be formulated as an MILP!

# A Simple Markov Chain Example

- Suppose that you live in a strange city where the weather is either sunny (S) or rainy (R), and you can forecast weather tomorrow based on weather today (independent of yesterday).

# A Simple Markov Chain Example

- Suppose that you live in a strange city where the weather is either sunny (S) or rainy (R), and you can forecast weather tomorrow based on weather today (independent of yesterday).
- Based on past data, you have the following probabilities:
  $\mathbb{P}(\text{ tomorrow is S } | \text{ today is S }) = 0.8$
  $\mathbb{P}(\text{ tomorrow is R } | \text{ today is S }) = 0.2$
  $\mathbb{P}(\text{ tomorrow is S } | \text{ today is R }) = 0.3$
  $\mathbb{P}(\text{ tomorrow is R } | \text{ today is R }) = 0.7$

# A Simple Markov Chain Example

- Suppose that you live in a strange city where the weather is either sunny (S) or rainy (R), and you can forecast weather tomorrow based on weather today (independent of yesterday).
- Based on past data, you have the following probabilities:
  $\mathbb{P}(\text{ tomorrow is S } | \text{ today is S }) = 0.8$
  $\mathbb{P}(\text{ tomorrow is R } | \text{ today is S }) = 0.2$
  $\mathbb{P}(\text{ tomorrow is S } | \text{ today is R }) = 0.3$
  $\mathbb{P}(\text{ tomorrow is R } | \text{ today is R }) = 0.7$
- What is the probability that Nov.19 will be R given that Nov.17 is S?

# A Simple Markov Chain Example

- Suppose that you live in a strange city where the weather is either sunny (S) or rainy (R), and you can forecast weather tomorrow based on weather today (independent of yesterday).
- Based on past data, you have the following probabilities:
  $\mathbb{P}(\text{ tomorrow is S } | \text{ today is S }) = 0.8$
  $\mathbb{P}(\text{ tomorrow is R } | \text{ today is S }) = 0.2$
  $\mathbb{P}(\text{ tomorrow is S } | \text{ today is R }) = 0.3$
  $\mathbb{P}(\text{ tomorrow is R } | \text{ today is R }) = 0.7$
- What is the probability that Nov.19 will be R given that Nov.17 is S?
- What is the long term probability of observing a R day?

# A Simple Markov Chain Example

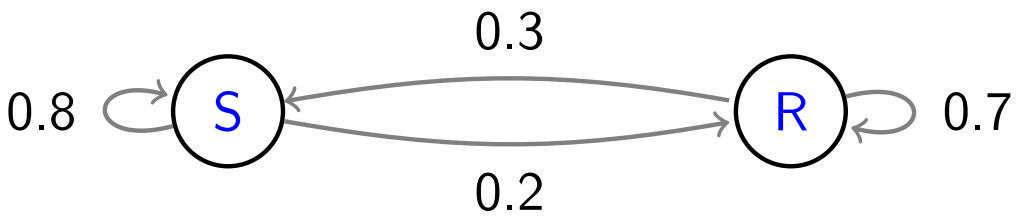
- There are two states in this Markov chain: S and R.

# A Simple Markov Chain Example

- There are two states in this Markov chain: S and R.
- Here is the associated *transition probability matrix* (think of rows as today and columns as tomorrow): $\boldsymbol{M} = \begin{matrix} \\ \text{S} \\ \text{R} \end{matrix} \begin{matrix} \begin{matrix} \text{S} & \text{R} \end{matrix} \\ \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix} \end{matrix}$

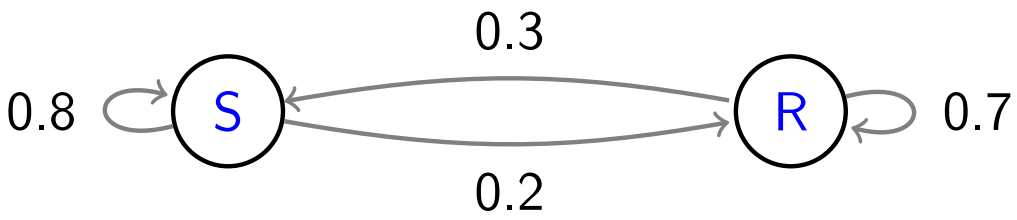
# A Simple Markov Chain Example

- There are two states in this Markov chain: S and R.
- Here is the associated *transition probability matrix* (think of rows as today and columns as tomorrow): $\boldsymbol{M} = \begin{matrix} \\ \text{S} \\ \text{R} \end{matrix} \begin{matrix} \begin{matrix} \text{S} & \text{R} \end{matrix} \\ \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix} \end{matrix}$
- Here is the associated *transition probability diagram*:



# A Simple Markov Chain Example

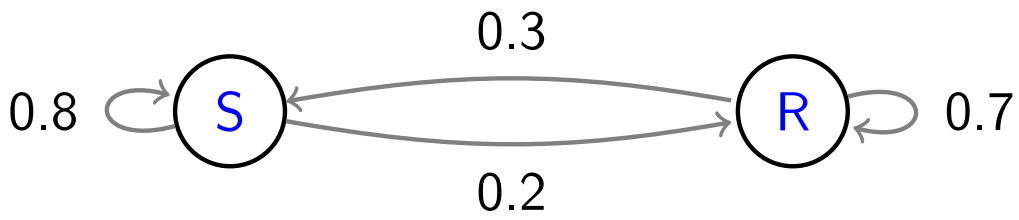
- There are two states in this Markov chain: S and R.
- Here is the associated *transition probability matrix* (think of rows as today and columns as tomorrow): $\boldsymbol{M} = \begin{matrix} \\ \text{S} \\ \text{R} \end{matrix} \begin{matrix} \begin{matrix} \text{S} & \text{R} \end{matrix} \\ \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix} \end{matrix}$
- Here is the associated *transition probability diagram*:




- What is the probability that Nov.19 will be R given that Nov.17 is S?
  $\mathbb{P}(\text{ Nov.19 is R } | \text{ Nov.17 is S }) = \boldsymbol{M}^2_{SR} = 0.3$

# A Simple Markov Chain Example

- There are two states in this Markov chain: S and R.
- Here is the associated *transition probability matrix* (think of rows as today and columns as tomorrow): $\boldsymbol{M} = \begin{matrix} \\ \mathrm{S} \\ \mathrm{R} \end{matrix} \begin{matrix} \begin{matrix} \mathrm{S} & \mathrm{R} \end{matrix} \\ \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix} \end{matrix}$
- Here is the associated *transition probability diagram*:




- What is the probability that Nov.19 will be R given that Nov.17 is S?
  $\mathbb{P}(\text{ Nov.19 is R } | \text{ Nov.17 is S }) = \boldsymbol{M}^2_{SR} = 0.3$
- What is the long term probability of observing a R day?

$$\boldsymbol{M}^{\infty} = \begin{matrix} \\ \mathrm{S} \\ \mathrm{R} \end{matrix} \begin{matrix} \begin{matrix} \mathrm{S} & \mathrm{R} \end{matrix} \\ \begin{bmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \end{bmatrix} \end{matrix}$$

# Another Markov Chain Example

- Suppose that you move to another strange city where the weather is either sunny (S) or rainy (R), but you can forecast weather tomorrow based on weather today AND yesterday.

# Another Markov Chain Example

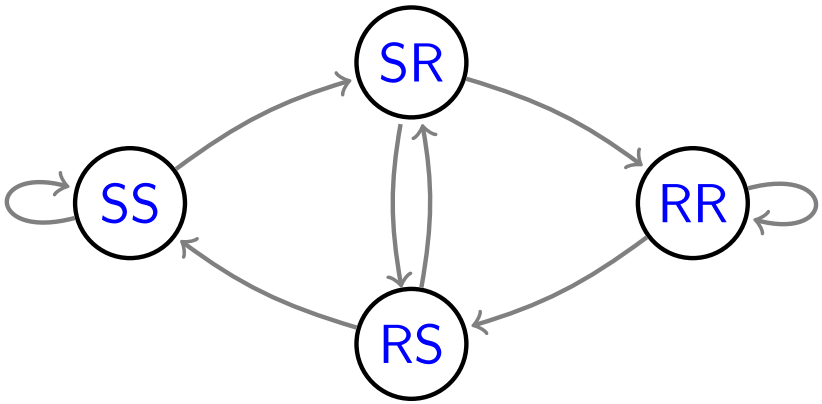
- Suppose that you move to another strange city where the weather is either sunny (S) or rainy (R), but you can forecast weather tomorrow based on weather today AND yesterday.
- Can this process be a Markov chain?

# Another Markov Chain Example

- Suppose that you move to another strange city where the weather is either sunny (S) or rainy (R), but you can forecast weather tomorrow based on weather today AND yesterday.
- Can this process be a Markov chain?
  - Yes! The states are SS, SR, RS and RR representing the weather (yesterday, today).
  - Based on the weather (yesterday, today), we can analyze the transition to (today, tomorrow).

# Another Markov Chain Example

- Suppose that you move to another strange city where the weather is either sunny (S) or rainy (R), but you can forecast weather tomorrow based on weather today AND yesterday.
- Can this process be a Markov chain?
  - Yes! The states are SS, SR, RS and RR representing the weather (yesterday, today).
  - Based on the weather (yesterday, today), we can analyze the transition to (today, tomorrow).
- A possible transition diagram looks like this:



# Discrete-Time Markov Chains (DTMC)

- A stochastic process is said to have the *Markovian* or *memoryless* property if the next state is a function of the current state and NOT a function of the past states.

# Discrete-Time Markov Chains (DTMC)

- A stochastic process is said to have the *Markovian* or *memoryless* property if the next state is a function of the current state and NOT a function of the past states.
- A DTMC is defined over a set of states in which the state transitions occur at discrete time intervals (e.g. a second, a day, a year, ...).

# Discrete-Time Markov Chains (DTMC)

- A stochastic process is said to have the *Markovian* or *memoryless* property if the next state is a function of the current state and NOT a function of the past states.
- A DTMC is defined over a set of states in which the state transitions occur at discrete time intervals (e.g. a second, a day, a year, ...).
- A DTMC is characterized by a transition probability matrix.
  - All entries are nonnegative, row sums are equal to 1.

# Discrete-Time Markov Chains (DTMC)

- A stochastic process is said to have the *Markovian* or *memoryless* property if the next state is a function of the current state and NOT a function of the past states.
- A DTMC is defined over a set of states in which the state transitions occur at discrete time intervals (e.g. a second, a day, a year, ...).
- A DTMC is characterized by a transition probability matrix.
  - All entries are nonnegative, row sums are equal to 1.
- Given an initial state of a Markov chain, one might be interested in questions of the following sort:
  - What is the probability of being in a certain state after a specified number of transitions?
  - What is the probability of being in a certain state on the long run?
  - What is the probability of being "absorbed" in a certain state eventually?

# Discrete-Time Markov Chains (DTMC)

- A stochastic process is said to have the *Markovian* or *memoryless* property if the next state is a function of the current state and NOT a function of the past states.
- A DTMC is defined over a set of states in which the state transitions occur at discrete time intervals (e.g. a second, a day, a year, ...).
- A DTMC is characterized by a transition probability matrix.
  - All entries are nonnegative, row sums are equal to 1.
- Given an initial state of a Markov chain, one might be interested in questions of the following sort:
  - What is the probability of being in a certain state after a specified number of transitions?
  - What is the probability of being in a certain state on the long run?
  - What is the probability of being "absorbed" in a certain state eventually?
- Relevance to Time Machines: Transition from one genotype to another under the effect of an antibiotic is modelled as a DTMC!

# Outline

# Antibiotics Time Machine

- Antibiotic resistance is an alarming global healthcare issue.

# Antibiotics Time Machine

- Antibiotic resistance is an alarming global healthcare issue.
- *Antibiotics Time Machine* is an important problem to understand this issue and how it can be reversed.

# Antibiotics Time Machine

- Antibiotic resistance is an alarming global healthcare issue.
- *Antibiotics Time Machine* is an important problem to understand this issue and how it can be reversed.
- **Idea**: Apply known antibiotics in a way that the resistance can be reversed *so we go back in time*.
  - Mira et al. (2015, 2017); Nichol et al. (2018, 2019); Maltas et al. (2019, 2021, 2023); Aulin et al. (2021); Batra et al. (2021); ...

# Antibiotics Time Machine

- We are interested in a fitness landscape with $a$ many alleles/genes and denote a genotype/state by a vector $\boldsymbol{g} \in \{0,1\}^a$.

# Antibiotics Time Machine

- We are interested in a fitness landscape with $a$ many alleles/genes and denote a genotype/state by a vector $\boldsymbol{g} \in \{0,1\}^a$.
- Each allele $h \in \{1,\ldots,a\}$ in $\boldsymbol{g}$ can either be mutated ($g_h = 1$) or unmutated ($g_h = 0$). Clearly, there are $d := 2^a$ states.

# Antibiotics Time Machine

- We are interested in a fitness landscape with $a$ many alleles/genes and denote a genotype/state by a vector $\boldsymbol{g} \in \{0,1\}^a$.
- Each allele $h \in \{1,\ldots,a\}$ in $\boldsymbol{g}$ can either be mutated ($g_h = 1$) or unmutated ($g_h = 0$). Clearly, there are $d := 2^a$ states.
- The special genotype $\mathbf{0}$ is called the "wild type".

# Antibiotics Time Machine

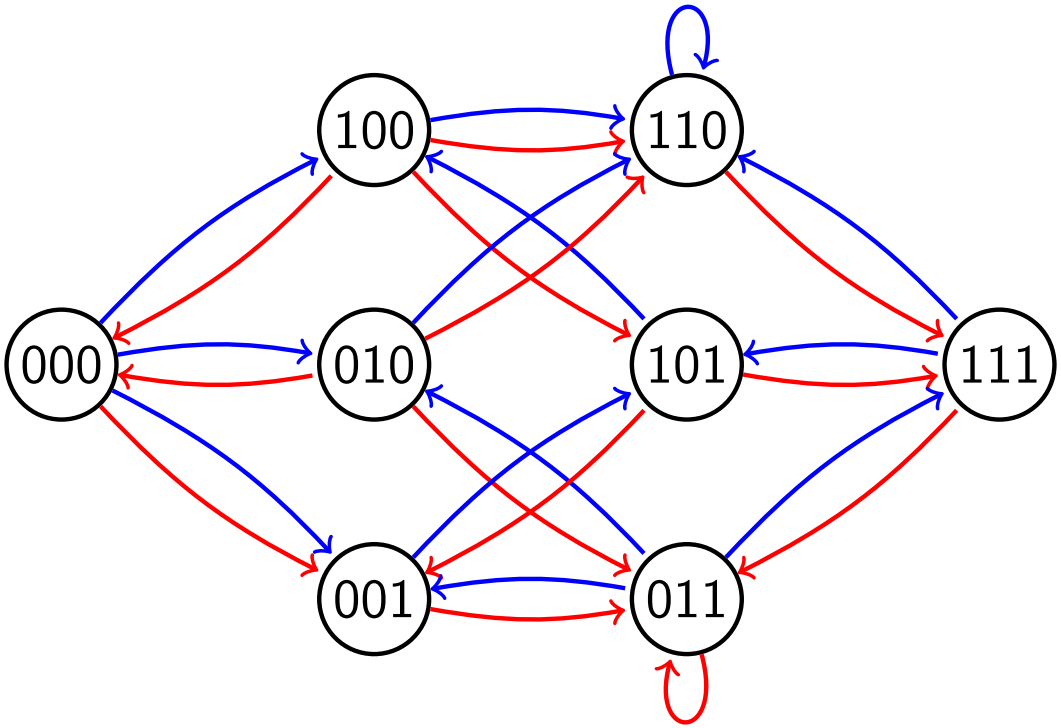
- We are interested in a fitness landscape with $a$ many alleles/genes and denote a genotype/state by a vector $\boldsymbol{g} \in \{0,1\}^a$.
- Each allele $h \in \{1,\ldots,a\}$ in $\boldsymbol{g}$ can either be mutated ($g_h = 1$) or unmutated ($g_h = 0$). Clearly, there are $d := 2^a$ states.
- The special genotype $\mathbf{0}$ is called the "wild type".
- If an antibiotic $k$ is administered at a certain state, then the transition to a new state is governed by a probability matrix $\boldsymbol{M^k} \in \mathbb{R}^{d \times d}$.
  - Suppose, for now, that $\boldsymbol{M^k}$ matrices are certain.

# Antibiotics Time Machine

- We are interested in a fitness landscape with $a$ many alleles/genes and denote a genotype/state by a vector $\boldsymbol{g} \in \{0,1\}^a$.
- Each allele $h \in \{1,\ldots,a\}$ in $\boldsymbol{g}$ can either be mutated ($g_h = 1$) or unmutated ($g_h = 0$). Clearly, there are $d := 2^a$ states.
- The special genotype $\mathbf{0}$ is called the "wild type".
- If an antibiotic $k$ is administered at a certain state, then the transition to a new state is governed by a probability matrix $\boldsymbol{M^k} \in \mathbb{R}^{d \times d}$.
  - Suppose, for now, that $\boldsymbol{M^k}$ matrices are certain.
- The aim is to maximize the probability of going from a certain state to the wild type in $N$ steps using any sequence of $K$ drugs.
  - Suppose, for now, that the sequence should be determined at the beginning.

# An Example with $a = 3$ and $K = 2$ Drugs

- Think of each color (Red and Blue) as a different antibiotic/drug.
- Observe that a single drug is not enough to go from state 111 to state 000.
- If $N = 3$ steps are allowed, administering Blue, Blue, Red or Red, Blue, Red gives a positive probability of going from 111 to 000.



# Problem Definition: *Static Deterministic* Version

- Let $\boldsymbol{p} \in \mathbb{R}^d$ ($\boldsymbol{q} \in \mathbb{R}^d$) be the given initial (desired) probability distribution vectors.
- Let $\mathcal{M} := \{\boldsymbol{M^k}\}_{k=1}^{K} \subseteq \mathbb{R}^{d \times d}$ be GIVEN probability matrices for each antibiotic $k$.

# Problem Definition: *Static Deterministic* Version

- Let $\boldsymbol{p} \in \mathbb{R}^d$ ($\boldsymbol{q} \in \mathbb{R}^d$) be the given initial (desired) probability distribution vectors.
- Let $\mathcal{M} := \{\boldsymbol{M^k}\}_{k=1}^K \subseteq \mathbb{R}^{d \times d}$ be GIVEN probability matrices for each antibiotic $k$.
- Consider the optimization problem

$$\max_{\boldsymbol{T}} \boldsymbol{p}^\top \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

# Problem Definition: *Static Deterministic* Version

- Let $\boldsymbol{p} \in \mathbb{R}^d$ ($\boldsymbol{q} \in \mathbb{R}^d$) be the given initial (desired) probability distribution vectors.
- Let $\mathcal{M} := \{\boldsymbol{M^k}\}_{k=1}^K \subseteq \mathbb{R}^{d \times d}$ be GIVEN probability matrices for each antibiotic $k$.
- Consider the optimization problem

$$\max_{\boldsymbol{T}} \boldsymbol{p}^\top \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- This is a "nonlinear combinatorial optimization" problem since
  - it contains the multiplication of $N$ variable matrices
  - the variable matrices $\boldsymbol{T_n}$'s must be selected from a finite collection.

# Problem Definition: *Static Deterministic* Version

- Let $\boldsymbol{p} \in \mathbb{R}^d$ ($\boldsymbol{q} \in \mathbb{R}^d$) be the given initial (desired) probability distribution vectors.
- Let $\mathcal{M} := \{\boldsymbol{M^k}\}_{k=1}^K \subseteq \mathbb{R}^{d \times d}$ be GIVEN probability matrices for each antibiotic $k$.
- Consider the optimization problem

$$\max_{\boldsymbol{T}} \boldsymbol{p}^\top \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- This is a "nonlinear combinatorial optimization" problem since
  - it contains the multiplication of $N$ variable matrices
  - the variable matrices $\boldsymbol{T_n}$'s must be selected from a finite collection.
- This version is *deterministic* since the antibiotic matrices are certain.

# Problem Definition: *Static Deterministic* Version

- Let $\boldsymbol{p} \in \mathbb{R}^d$ ($\boldsymbol{q} \in \mathbb{R}^d$) be the given initial (desired) probability distribution vectors.
- Let $\mathcal{M} := \{\boldsymbol{M^k}\}_{k=1}^K \subseteq \mathbb{R}^{d \times d}$ be GIVEN probability matrices for each antibiotic $k$.
- Consider the optimization problem

$$\max_{\boldsymbol{T}} \boldsymbol{p}^\top \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- This is a "nonlinear combinatorial optimization" problem since
  - it contains the multiplication of $N$ variable matrices
  - the variable matrices $\boldsymbol{T_n}$'s must be selected from a finite collection.
- This version is *deterministic* since the antibiotic matrices are certain.
- This version is *static* since the antibiotic sequence is decided at the beginning (out of $K^N$ possible such sequences).

# Literature: *Static Deterministic* Version

$$\max_{\boldsymbol{T}} \boldsymbol{p}^{\top} \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

# Literature: *Static Deterministic* Version

$$\max_{\boldsymbol{T}} \boldsymbol{p}^{\top} \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- The problem is shown to be NP-Hard (Tran and Yang, 2017).

# Literature: *Static Deterministic* Version

$$\max_{\boldsymbol{T}} \boldsymbol{p}^{\top} \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- The problem is shown to be NP-Hard (Tran and Yang, 2017).
- There is only limited attention from the optimization community (except Wu and He, 2018).

# Literature: *Static Deterministic* Version

$$\max_{\boldsymbol{T}} \boldsymbol{p}^{\top} \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- The problem is shown to be NP-Hard (Tran and Yang, 2017).
- There is only limited attention from the optimization community (except Wu and He, 2018).
- Existing solution methods were based on complete enumeration (e.g. Mira et al., 2015).

# Literature: *Static Deterministic* Version

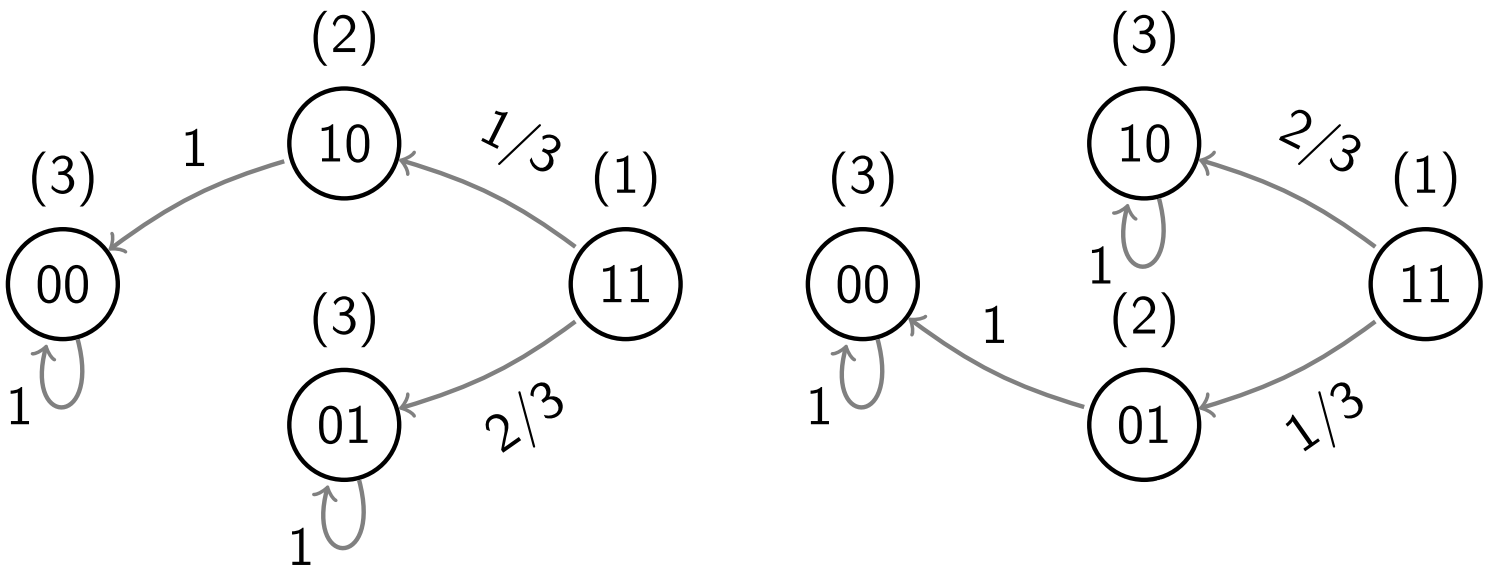
$$\max_{\boldsymbol{T}} \boldsymbol{p}^{\top} \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

- The problem is shown to be NP-Hard (Tran and Yang, 2017).
- There is only limited attention from the optimization community (except Wu and He, 2018).
- Existing solution methods were based on complete enumeration (e.g. Mira et al., 2015).
- Recently, a mixed-integer linear programming (MILP) formulation was proposed (Kocuk, 2022), which outperforms complete enumeration (as expected).

# Outline

# An Example with $a = 2$, $K = 2$ and $N = 2$ (initial $= 11$)

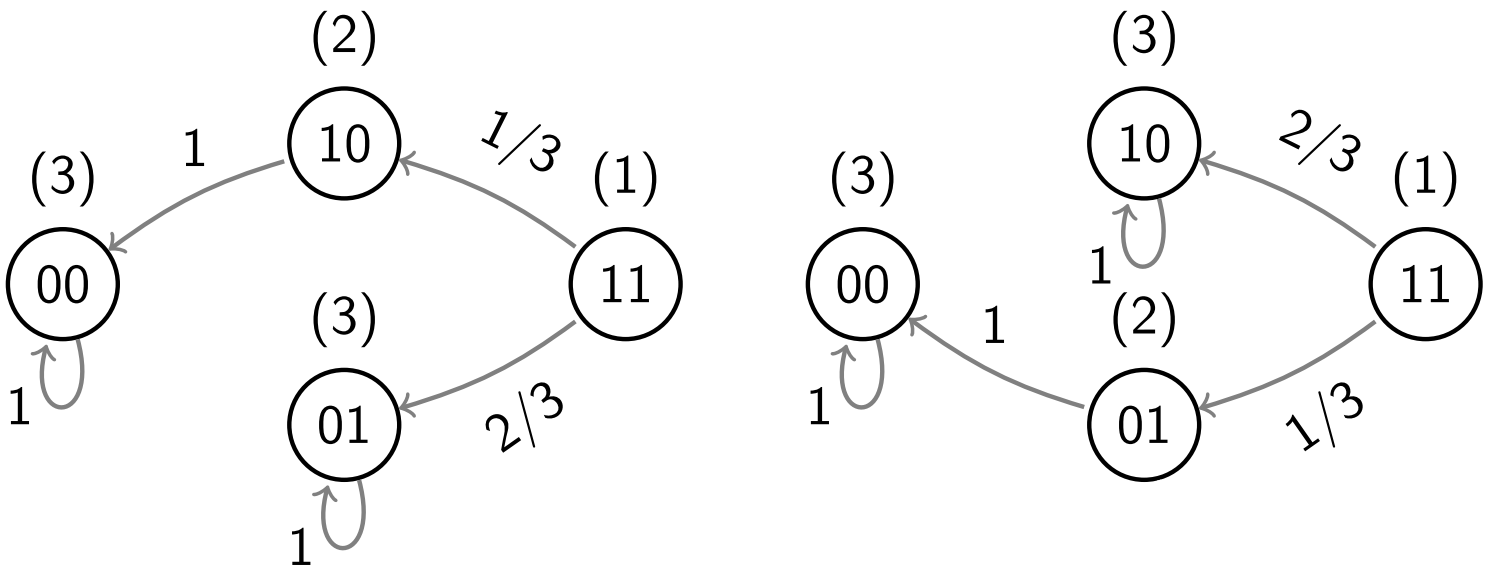



(a) Antibiotic I.

(b) Antibiotic II.

Figure: Markov chains corresponding to two antibiotics.

# An Example with $a = 2$, $K = 2$ and $N = 2$ (initial = 11)




(a) Antibiotic I.

(b) Antibiotic II.

Figure: Markov chains corresponding to two antibiotics.

*Static*: 4 possible sequences
Optimal Value: 2/3
Optimal Sequence: I-II or II-I

# Problem Definition

## Problem (Static Deterministic Version)

Given $K$ antibiotics with their transition probability matrices $\boldsymbol{M^k}$, $k = 1, \ldots, K$, the length of the treatment plan $N$ and the initial distribution $\boldsymbol{p}$, determine a *sequence* of antibiotics to be applied so that the probability of reaching the desired final distribution $\boldsymbol{q}$ is maximized.

# Transition Probability Matrix Computation

- We compute the transition probabilities as a function of the measured **growth rates** $\boldsymbol{\omega} \in \mathbb{R}_{+}^{d}$ of genotypes under each antibiotic (Mira et al., 2015).

# Transition Probability Matrix Computation

- We compute the transition probabilities as a function of the measured **growth rates** $\boldsymbol{\omega} \in \mathbb{R}_+^d$ of genotypes under each antibiotic (Mira et al., 2015).
- Due to the widely accepted assumption of Gillespie (1983, 1984) called *Strong Selection Weak Mutation*, the transitions between two genotypes $\boldsymbol{g}$ and $\boldsymbol{g'}$ can occur only if a single allele is different. Let us denote the set of such pairs as $J$.

# Transition Probability Matrix Computation

- We compute the transition probabilities as a function of the measured **growth rates** $\boldsymbol{\omega} \in \mathbb{R}_+^d$ of genotypes under each antibiotic (Mira et al., 2015).
- Due to the widely accepted assumption of Gillespie (1983, 1984) called *Strong Selection Weak Mutation*, the transitions between two genotypes $\boldsymbol{g}$ and $\boldsymbol{g'}$ can occur only if a single allele is different. Let us denote the set of such pairs as $J$.
- For a pair of neighbors $(j, j') \in J$ for antibiotic $k$, the transition probability from genotype $j$ to genotype $j'$ is

$$M_{j,j'}^k(\boldsymbol{\omega}) \begin{cases} > 0 & \text{if } \omega_{j'} > \omega_j \\ = 0 & \text{otherwise} \end{cases}.$$

# Transition Probability Matrix Computation

- We compute the transition probabilities as a function of the measured **growth rates** $\boldsymbol{\omega} \in \mathbb{R}_+^d$ of genotypes under each antibiotic (Mira et al., 2015).
- Due to the widely accepted assumption of Gillespie (1983, 1984) called *Strong Selection Weak Mutation*, the transitions between two genotypes $\boldsymbol{g}$ and $\boldsymbol{g'}$ can occur only if a single allele is different. Let us denote the set of such pairs as $J$.
- For a pair of neighbors $(j, j') \in J$ for antibiotic $k$, the transition probability from genotype $j$ to genotype $j'$ is

$$M_{j,j'}^k(\boldsymbol{\omega}) \begin{cases} > 0 & \text{if } \omega_{j'} > \omega_j \\ = 0 & \text{otherwise} \end{cases}.$$

- Following Mira et al. (2015), we consider two models regarding how the precise transition probabilities are computed.

# Transition Probability Matrix Computation

- Correlated Probability Model (CPM): In this model, the elements of the transition probability matrix are computed as

$$C_{j,j'}(\boldsymbol{\omega}) := \frac{\max\{0, \omega_{j'} - \omega_j\}}{\sum_{j'':(j'',j)\in J} \max\{0, \omega_{j''} - \omega_j\}}, \quad (j, j') \in J.$$

- Equal Probability Model (EPM): In this model, the elements of the transition probability matrix are computed as

$$E_{j,j'}(\boldsymbol{\omega}) := \frac{\mathbf{1}(\omega_{j'} > \omega_j)}{\sum_{j'':(j'',j)\in J} \mathbf{1}(\omega_{j''} > \omega_j)}, \quad (j, j') \in J.$$

# Transition Probability Matrix Computation

- Correlated Probability Model (CPM): In this model, the elements of the transition probability matrix are computed as

$$C_{j,j'}(\boldsymbol{\omega}) := \frac{\max\{0, \omega_{j'} - \omega_j\}}{\sum_{j'':(j'',j)\in J} \max\{0, \omega_{j''} - \omega_j\}}, \quad (j, j') \in J.$$

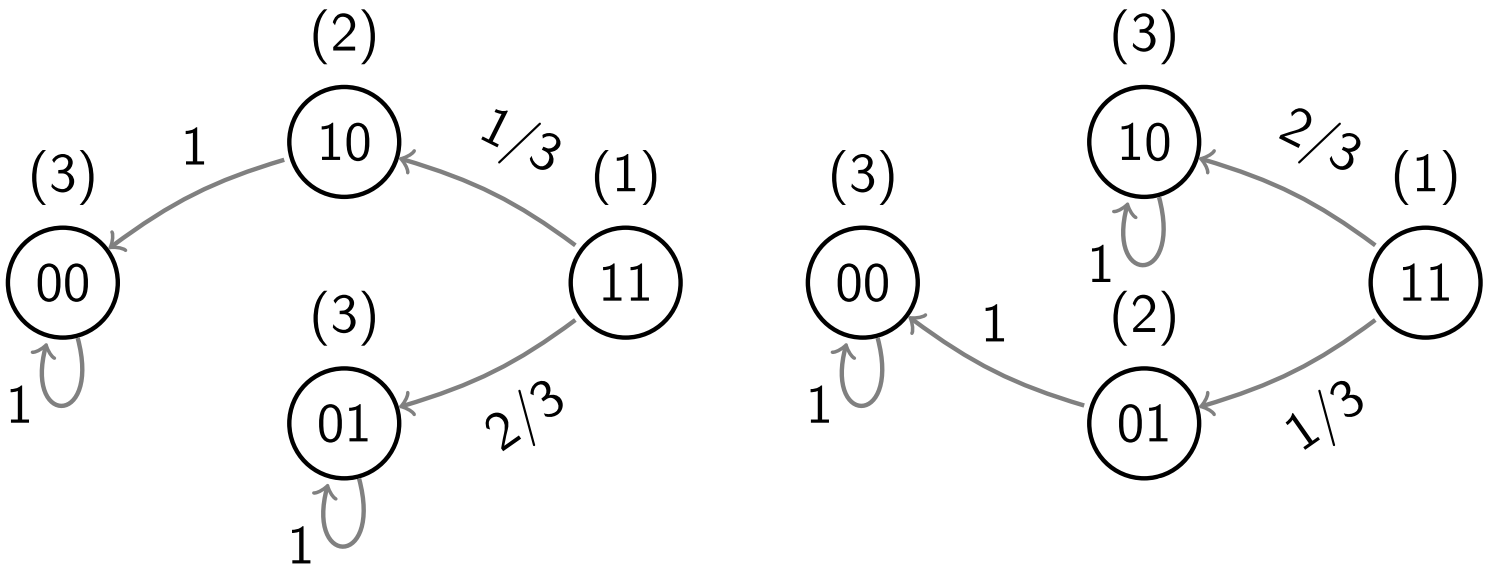
- Equal Probability Model (EPM): In this model, the elements of the transition probability matrix are computed as

$$E_{j,j'}(\boldsymbol{\omega}) := \frac{\mathbf{1}(\omega_{j'} > \omega_j)}{\sum_{j'':(j'',j)\in J} \mathbf{1}(\omega_{j''} > \omega_j)}, \quad (j, j') \in J.$$

For each antibiotic $k$, we will denote its growth rate vector as $\boldsymbol{\omega}^{\boldsymbol{k}}$, and set the probability transition matrix $\boldsymbol{M}^{\boldsymbol{k}} = \boldsymbol{C}(\boldsymbol{\omega}^{\boldsymbol{k}})$ if CPM is used and $\boldsymbol{M}^{\boldsymbol{k}} = \boldsymbol{E}(\boldsymbol{\omega}^{\boldsymbol{k}})$ if EPM is used.

# Transition Probability Matrix Computation

In this example with $a = 2$ and $K = 2$, the CPM is used to compute the transition probabilities (the numbers near nodes are the growth rates).




(a) Antibiotic I.

(b) Antibiotic II.

Figure: Markov chains corresponding to two antibiotics.

# Static Optimization

**Problem (Static Deterministic Version)**

Given $K$ antibiotics with their transition probability matrices $\boldsymbol{M^k}$, $k = 1, \ldots, K$, the length of the treatment plan $N$ and the initial distribution $\boldsymbol{p}$, determine a *sequence* of antibiotics to be applied so that the probability of reaching the desired final distribution $\boldsymbol{q}$ is maximized.

# Static Optimization

**Problem (Static Deterministic Version)**

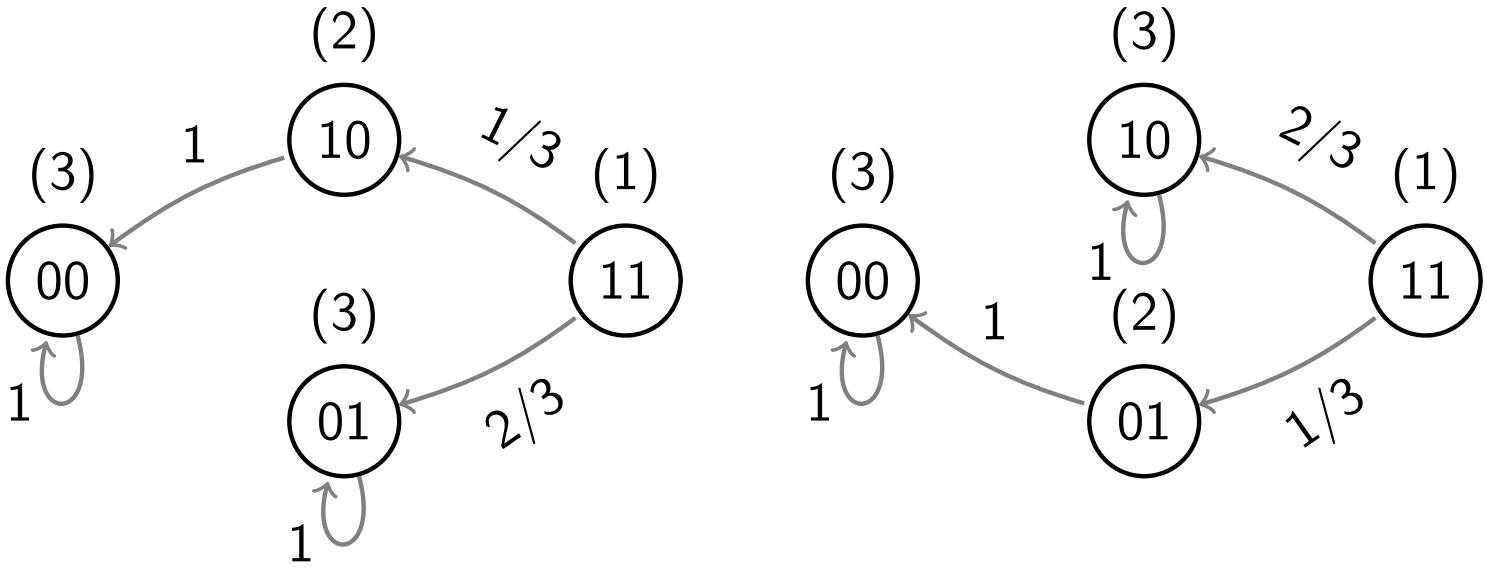
Given $K$ antibiotics with their transition probability matrices $\boldsymbol{M^k}$, $k = 1, \ldots, K$, the length of the treatment plan $N$ and the initial distribution $\boldsymbol{p}$, determine a *sequence* of antibiotics to be applied so that the probability of reaching the desired final distribution $\boldsymbol{q}$ is maximized.

Nonlinear Formulation:

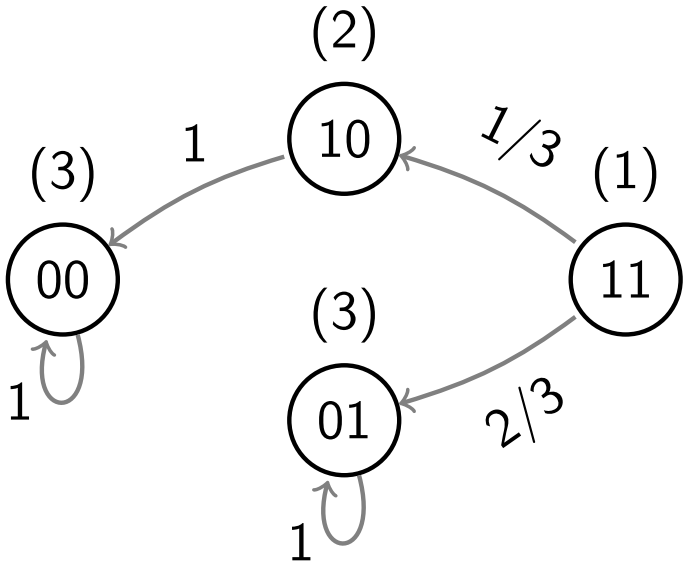
$$\max \boldsymbol{p}^\top \left( \prod_{n=1}^{N} \boldsymbol{T_n} \right) \boldsymbol{q}$$

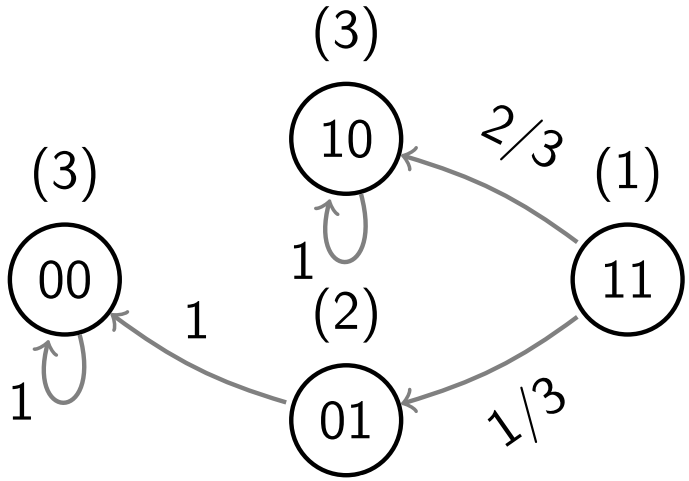
$$\text{s.t. } \boldsymbol{T_n} \in \mathcal{M} := \{\boldsymbol{M^k} : k = 1, \ldots, K\} \qquad n = 1, \ldots, N.$$

# An Example with $a = 2$, $K = 2$ and $N = 2$ (initial = 11)



# An Example with $a = 2$, $K = 2$ and $N = 2$ (initial $= 11$)

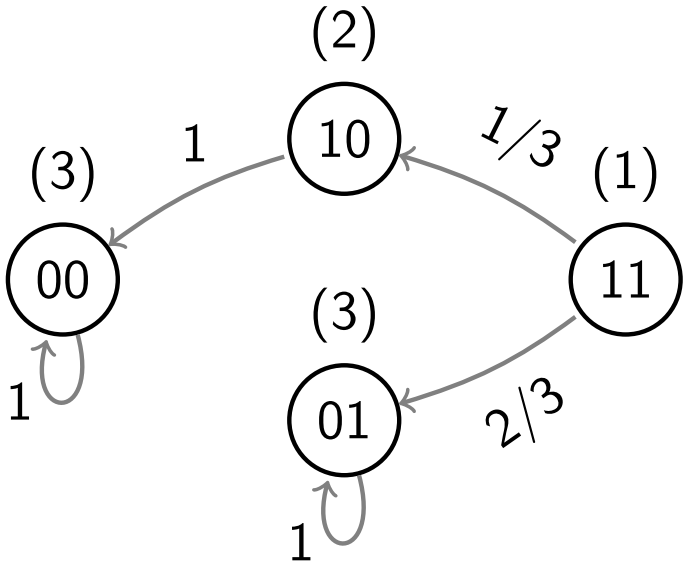


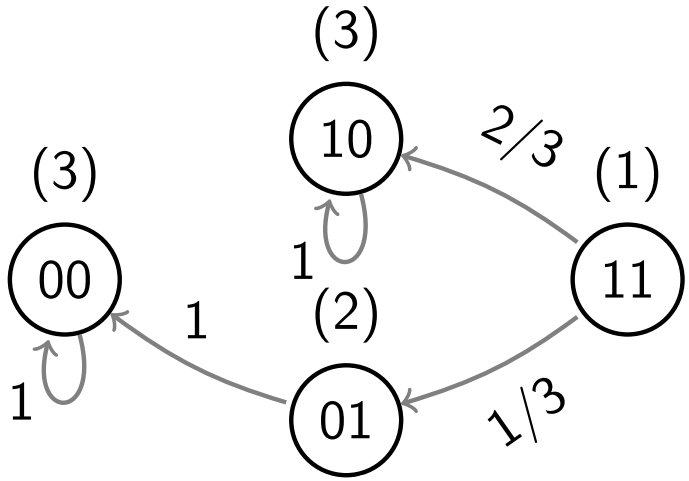




$$M^{\mathrm{I}} = \begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1/3 & 2/3 & 0 \end{bmatrix} \end{array},$$

$$M^{\mathrm{II}} = \begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2/3 & 1/3 & 0 \end{bmatrix} \end{array}.$$

# An Example with $a = 2$, $K = 2$ and $N = 2$ (initial $= 11$)




$$M^{\mathrm{I}} = \begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1/3 & 2/3 & 0 \end{bmatrix} \end{array}, \qquad M^{\mathrm{II}} = \begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2/3 & 1/3 & 0 \end{bmatrix} \end{array}.$$

$$\textbf{SOLVE} \ \max \left\{ \boldsymbol{p}^{\top} \left( \boldsymbol{T}_1 \boldsymbol{T}_2 \right) \boldsymbol{q} : \boldsymbol{T}_1, \boldsymbol{T}_2 \in \{ M^{\mathrm{I}}, M^{\mathrm{II}} \} \right\},$$

where $\boldsymbol{p}^{\top} = \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$ and $\boldsymbol{q}^{\top} = \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$.

# Static Optimization: MILP Formulation (Kocuk, 2022)

- The key idea is to define two sets of decisions variables:
  - $x_{n,k}$: one if antibiotic $k$ is applied in step $n$ and zero otherwise
  - $\boldsymbol{u_n} \in \mathbb{R}_+^d$: the probability distribution after $n$ transitions.

# Static Optimization: MILP Formulation (Kocuk, 2022)

- The key idea is to define two sets of decisions variables:
  - $x_{n,k}$: one if antibiotic $k$ is applied in step $n$ and zero otherwise
  - $\boldsymbol{u_n} \in \mathbb{R}^d_+$: the probability distribution after $n$ transitions.

- To ensure exactly one antibiotic selection per step, we must have

$$\sum_{k=1}^{K} x_{n,k} = 1 \qquad \forall n.$$

# Static Optimization: MILP Formulation (Kocuk, 2022)

- The key idea is to define two sets of decisions variables:
  - $x_{n,k}$: one if antibiotic $k$ is applied in step $n$ and zero otherwise
  - $\boldsymbol{u_n} \in \mathbb{R}^d_+$: the probability distribution after $n$ transitions.

- To ensure exactly one antibiotic selection per step, we must have

$$\sum_{k=1}^{K} x_{n,k} = 1 \qquad \forall n.$$

- Notice that these variables satisfy the recursion

$$\boldsymbol{u_n}^\top = \sum_{k=1}^{K} \boldsymbol{u_{n-1}}^\top \boldsymbol{M^k} x_{n,k} \qquad \forall n,$$

where $\boldsymbol{u_0} = \boldsymbol{p}$.

# Static Optimization: MILP Formulation (Kocuk, 2022)

- The key idea is to define two sets of decisions variables:
  - $x_{n,k}$: one if antibiotic $k$ is applied in step $n$ and zero otherwise
  - $\boldsymbol{u_n} \in \mathbb{R}^d_+$: the probability distribution after $n$ transitions.

- To ensure exactly one antibiotic selection per step, we must have

$$\sum_{k=1}^{K} x_{n,k} = 1 \qquad \forall n.$$

- Notice that these variables satisfy the recursion

$$\boldsymbol{u_n}^\top = \sum_{k=1}^{K} \boldsymbol{u_{n-1}}^\top \boldsymbol{M^k} x_{n,k} \qquad \forall n,$$

  where $\boldsymbol{u_0} = \boldsymbol{p}$.

- This relation is linearized using a disjunctive formulation by defining copy variables $\boldsymbol{v_{n,k}} \in \mathbb{R}^d_+$, which take value $\boldsymbol{u_{n-1}}$ if $x_{n,k} = 1$.

# Static Optimization: MILP Formulation

Let $\boldsymbol{e}$ be the vector of ones.

$$\max_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}} \boldsymbol{q}^\top \boldsymbol{u_N} \qquad \text{(maximize probability)}$$

$$\text{s.t. } \boldsymbol{u_0} = \boldsymbol{p} \qquad \text{(initial condition)}$$

$$\sum_{k=1}^{K} \boldsymbol{v_{n-1,k}} = \boldsymbol{u_{n-1}} \qquad \forall n \qquad \text{(disjunction-1)}$$

$$\sum_{k=1}^{K} \boldsymbol{v_{n-1,k}}^\top \boldsymbol{M^k} = \boldsymbol{u_n}^\top \qquad \forall n \qquad \text{(disjunction-2)}$$

$$\boldsymbol{e}^\top \boldsymbol{v_{n-1,k}} = x_{n,k} \qquad \forall n, k \qquad \text{(disjunction-3)}$$

$$\sum_{k=1}^{K} x_{n,k} = 1 \qquad \forall n \qquad \text{(one drug/step)}$$

$$\boldsymbol{v_{n-1,k}} \in \mathbb{R}^d_+, \boldsymbol{u_n} \in \mathbb{R}^d_+, \; x_{n,k} \in \{0,1\} \qquad \forall n, k \qquad \text{(domains)}$$

"Proposition": For each $n$, our extended (disjunctive) formulation gives the convex hull of the recursive relation.

# Outline

# Computational Setting

- We use the experimental growth rate data from Mira et al. (2015) to obtain the transition probability matrices ($d = 16$ genotypes, $K = 15$ different antibiotics).

# Computational Setting

- We use the experimental growth rate data from Mira et al. (2015) to obtain the transition probability matrices ($d = 16$ genotypes, $K = 15$ different antibiotics).
- We use a 64-bit workstation with Intel Core i7 CPU (2.60GHz) processors (16 GB RAM) using the Python programming language.

# Computational Setting

- We use the experimental growth rate data from Mira et al. (2015) to obtain the transition probability matrices ($d = 16$ genotypes, $K = 15$ different antibiotics).
- We use a 64-bit workstation with Intel Core i7 CPU (2.60GHz) processors (16 GB RAM) using the Python programming language.
- We utilize Gurobi 9 to solve the MILPs with the default settings (except the absolute optimality gap parameter is set to 0.001).

# EPM Results

| Initial | $N=4$ | $N=5$ | $N=6$ | $N=7$ | $N=8$ | $N=9$ | $N=12$ | $N=15$ |
|---|---|---|---|---|---|---|---|---|
| **1000** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| **0100** | 0.375 | 0.458 | 0.458 | 0.463 | 0.463 | 0.471 | 0.515 | 0.520 |
| **0010** | 0.500 | 0.500 | 0.500 | 0.512 | 0.512 | 0.515 | 0.520 | 0.532 |
| **0001** | 0.667 | 0.667 | 0.667 | 0.690 | 0.690 | 0.693 | 0.696 | 0.704 |
| **1100** | 0.389 | 0.389 | 0.458 | 0.458 | 0.463 | 0.463 | 0.479 | 0.520 |
| **1010** | 0.583 | 0.583 | 0.587 | 0.587 | 0.591 | 0.591 | 0.601 | 0.606 |
| **1001** | 0.667 | 0.667 | 0.690 | 0.690 | 0.693 | 0.693 | 0.700 | 0.704 |
| **0110** | 0.333 | 0.375 | 0.458 | 0.458 | 0.463 | 0.463 | 0.479 | 0.520 |
| **0101** | 0.458 | 0.458 | 0.463 | 0.463 | 0.471 | 0.479 | 0.515 | 0.526 |
| **0011** | 0.500 | 0.500 | 0.500 | 0.502 | 0.531 | 0.539 | 0.557 | 0.562 |
| **1110** | 0.333 | 0.333 | 0.375 | 0.458 | 0.458 | 0.463 | 0.479 | 0.515 |
| **1101** | 0.375 | 0.458 | 0.458 | 0.463 | 0.463 | 0.471 | 0.515 | 0.520 |
| **1011** | 0.333 | 0.389 | 0.417 | 0.458 | 0.458 | 0.475 | 0.481 | 0.515 |
| **0111** | 0.198 | 0.333 | 0.375 | 0.458 | 0.458 | 0.463 | 0.479 | 0.515 |
| **1111** | 0.333 | 0.375 | 0.458 | 0.458 | 0.463 | 0.463 | 0.479 | 0.520 |
| MILP(s) | 0.14 | 0.25 | 0.42 | 0.6 | 0.96 | 1.96 | 14.96 | 165.86 |
| BBNode | 4 | 43 | 168 | 448 | 1002 | 2380 | 22676 | 125473 |
| Enum(s) | 0.94 | 16.46 | 273.02 | $4 \cdot 10^3$ | $6 \cdot 10^4$ | $9 \cdot 10^5$ | $3 \cdot 10^9$ | $1 \cdot 10^{13}$ |

- Probabilities of going from initial state to wild type (0000) are reported.
- MILP (sec): Avg. CPU time of Gurobi (absolute opt. gap is set to 0.001).
- Enum (sec): Avg. (serial) time of the enumeration (estimated for $N > 6$).

# CPM Results

| Initial | $N = 4$ | $N = 5$ | $N = 6$ | $N = 7$ | $N = 8$ | $N = 9$ | $N = 12$ | $N = 15$ |
|---|---|---|---|---|---|---|---|---|
| **1000** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| **0100** | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **0010** | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 |
| **0001** | 0.592 | 0.726 | 0.726 | 0.729 | 0.729 | 0.729 | 0.731 | 0.733 |
| **1100** | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **1010** | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 | 0.715 |
| **1001** | 0.726 | 0.726 | 0.729 | 0.729 | 0.729 | 0.729 | 0.732 | 0.733 |
| **0110** | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **0101** | 0.612 | 0.612 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **0011** | 0.586 | 0.600 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **1110** | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **1101** | 0.592 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **1011** | 0.532 | 0.684 | 0.690 | 0.691 | 0.693 | 0.694 | 0.695 | 0.697 |
| **0111** | 0.600 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| **1111** | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 | 0.617 |
| MILP(s) | 0.12 | 0.19 | 0.41 | 0.63 | 1.01 | 1.42 | 10.21 | 147.29 |
| BBNode | 2 | 8 | 26 | 84 | 252 | 602 | 8093 | 97275 |
| Enum(s) | 0.94 | 16.46 | 272.12 | $4 \cdot 10^3$ | $6 \cdot 10^4$ | $9 \cdot 10^5$ | $3 \cdot 10^9$ | $1 \cdot 10^{13}$ |

- Probabilities of going from initial state to wild type (0000) are reported.
- MILP (sec): Avg. CPU time of Gurobi (absolute opt. gap is set to 0.001).
- Enum (sec): Avg. (serial) time of the enumeration (estimated for $N > 6$).