# Making smart decision under uncertainty

V. Leclère

October 10 2024



# Presentation Outline

# Presentation Outline

# Why is there uncertainty?

- Parameters might not be known exactly (there might be some error in the measurement);
- We want to take into account event that have yet to come (*e.g.*, weather), such event could be predicted but not with certainty;
- Data might be missing or corrupted, and we have to take a decision based on available data nonetheless;
- ...

# Can't we just look at different values of the parameters?

- A natural idea, when confronted with uncertainty, is to just look at various possible values of the uncertain parameters and optimize for each of them (sometimes called "scenario optimization").
- Unfortunately, this is not enough and can even be quite misleading.
- Let's take a first example:
  - You have 1000€ to trade on the stock market;
  - Consider a stock that has a 50% chance of going up by 10% and a 50% chance of going down by 10%;
  - Assume that, for a flat fee of 50€, you can buy or sell the stock;

  ➥ What happens if you just look at the two values and optimize for each?

# Can't we just look at different values of the parameters?

- A natural idea, when confronted with uncertainty, is to just look at various possible values of the uncertain parameters and optimize for each of them (sometimes called "scenario optimization").
- Unfortunately, this is not enough and can even be quite misleading.
- Let's take a first example:
  - You have 1000€ to trade on the stock market;
  - Consider a stock that has a 50% chance of going up by 10% and a 50% chance of going down by 10%;
  - Assume that, for a flat fee of 50€, you can buy or sell the stock;

  ➥ What happens if you just look at the two values and optimize for each?

# Common solution might not be relevant

Assume that you can produce 3 products $A$, $B$ and $C$.

- Assume total units produced is limited to 100, with a production cost of 10 per unit;
- Selling price of $A$ is 11 per unit, $B$ is 12 per unit and $C$ is 13 per unit;
- The demand for $A$ is unlimited, for $B$ and $C$ the total demand is 100, but the repartition is unknown – let denote $\xi$ the demand for $B$.

1. For a given $\xi$, write the optimization problem;
2. What is, for each $\xi$, the optimal production plan?
3. Is there a decision common to all $\xi$?
4. Assume now that $\xi$ take value 0 or 100 with equal probability, what is the optimal production plan? Comments?

# Common solution might not be relevant

Assume that you can produce 3 products $A$, $B$ and $C$.

- Assume total units produced is limited to 100, with a production cost of 10 per unit;
- Selling price of $A$ is 11 per unit, $B$ is 12 per unit and $C$ is 13 per unit;
- The demand for $A$ is unlimited, for $B$ and $C$ the total demand is 100, but the repartition is unknown – let denote $\xi$ the demand for $B$.

1. For a given $\xi$, write the optimization problem;
2. What is, for each $\xi$, the optimal production plan?
3. Is there a decision common to all $\xi$?
4. Assume now that $\xi$ take value 0 or 100 with equal probability, what is the optimal production plan? Comments?

# Flexibility is not valued

- Assume that you need to buy a certain amount of a given product. The current price is 100.
- You can buy now, or wait and buy at an unknown price $p \in [50, 200]$.
- A trader offers you, for 10, to have the option to buy, later, at the current price 100.

1. What is the optimal decision if you know the value of $p$?
2. Assume that $p$ can only take value 50 and 200 with equal probability, what is the optimal decision?

# Presentation Outline

# An optimization problem with uncertainty

Adding uncertainty $\xi$ in the mix

$$\begin{aligned} \min_{u_0} \quad & L(u_0, \xi) \\ s.t. \quad & g(u_0, \xi) \leq 0 \end{aligned}$$

Remarks:

- $\xi$ is unknown. Two main way of modelling it:
  - $\xi \in \Xi$ with a known uncertainty set $\Xi$, and a pessimistic approach. This is the robust optimization approach (RO).
  - $\xi$ is a random variable with known probability law. This is the Stochastic Programming approach (SP).
- Cost is not well defined.
  - RO : $\max_{\xi \in \Xi} L(u, \xi)$.
  - SP : $\mathbb{E}\left[L(u, \boldsymbol{\xi})\right]$.
- Constraints are not well defined.
  - RO : $g(u, \xi) \leq 0, \qquad \forall \xi \in \Xi$.
  - SP : $g(u, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P} - a.s.$.

# An optimization problem with uncertainty

Adding uncertainty $\xi$ in the mix

$$\begin{aligned} \min_{u_0} \quad & L(u_0, \xi) \\ s.t. \quad & g(u_0, \xi) \leq 0 \end{aligned}$$

Remarks:

- $\xi$ is unknown. Two main way of modelling it:
  - $\xi \in \Xi$ with a known uncertainty set $\Xi$, and a pessimistic approach. This is the robust optimization approach (RO).
  - $\xi$ is a random variable with known probability law. This is the Stochastic Programming approach (SP).
- Cost is not well defined.
  - RO : $\max_{\xi \in \Xi} L(u, \xi)$.
  - SP : $\mathbb{E}\left[L(u, \xi)\right]$.
- Constraints are not well defined.
  - RO : $g(u, \xi) \leq 0, \qquad \forall \xi \in \Xi$.
  - SP : $g(u, \xi) \leq 0, \qquad \mathbb{P} - a.s.$.

# An optimization problem with uncertainty

Adding uncertainty $\xi$ in the mix

$$\begin{aligned} \min_{u_0} \quad & L(u_0, \xi) \\ s.t. \quad & g(u_0, \xi) \leq 0 \end{aligned}$$

Remarks:

- $\xi$ is unknown. Two main way of modelling it:
  - $\xi \in \Xi$ with a known uncertainty set $\Xi$, and a pessimistic approach. This is the robust optimization approach (RO).
  - $\xi$ is a random variable with known probability law. This is the Stochastic Programming approach (SP).
- Cost is not well defined.
  - RO : $\max_{\xi \in \Xi} L(u, \xi)$.
  - SP : $\mathbb{E}\left[L(u, \boldsymbol{\xi})\right]$.
- Constraints are not well defined.
  - RO : $g(u, \xi) \leq 0, \qquad \forall \xi \in \Xi$.
  - SP : $g(u, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P} - a.s.$.

# The (deterministic) newsboy problem

In the 50's a boy would buy a stock $u$ of newspapers each morning at a cost $c$, and sell them all day long for a price $p$. The number of people interested in buying a paper during the day is $d$. We assume that $0 < c < p$.

How shall we model this ?

# The (deterministic) newsboy problem

In the 50's a boy would buy a stock $u$ of newspapers each morning at a cost $c$, and sell them all day long for a price $p$. The number of people interested in buying a paper during the day is $d$. We assume that $0 < c < p$.

How shall we model this ?

- Control $u \in \mathbb{R}^+$
- Cost $L(u) = cu - p\min(u, d)$

Leading to

$$\begin{aligned} \min_u \quad & cu - p\min(u, d) \\ s.t. \quad & u \geq 0 \end{aligned}$$

# The (stochastic) newsboy problem

Demand $d$ is unknown at time of purchasing. We model it as a random variable $\boldsymbol{d}$ with known law. Note that

- the control $u \in \mathbb{R}^+$ is deterministic
- the cost is a random variable (depending of $\boldsymbol{d}$). We choose to minimize its expectation.

# The (stochastic) newsboy problem

Demand $d$ is unknown at time of purchasing. We model it as a random variable $\boldsymbol{d}$ with known law. Note that

- the control $u \in \mathbb{R}^+$ is deterministic
- the cost is a random variable (depending of $\boldsymbol{d}$). We choose to minimize its expectation.

We consider the following problem

$$\begin{aligned} \min_{u} \quad & \mathbb{E}\left[cu - p\min(u, \boldsymbol{d})\right] \\ s.t. \quad & u \geq 0 \end{aligned}$$

How can we justify the expectation ?

# The (stochastic) newsboy problem

Demand $d$ is unknown at time of purchasing. We model it as a random variable $\boldsymbol{d}$ with known law. Note that

- the control $u \in \mathbb{R}^+$ is deterministic
- the cost is a random variable (depending of $\boldsymbol{d}$). We choose to minimize its expectation.

We consider the following problem

$$\begin{aligned} \min_{u} \quad & \mathbb{E}\left[cu - p\min(u, \boldsymbol{d})\right] \\ s.t. \quad & u \geq 0 \end{aligned}$$

How can we justify the expectation ?
By law of large number: the Newsboy is going to sell newspaper again and again. Then optimizing the sum over time of its gains is closely related to optimizing the expected gains.

# Alternative cost functions I

- When the cost $L(u, \xi)$ is random it might be natural to want to minimize its expectation $\mathbb{E}\left[L(u, \xi)\right]$.
- This is even justified if the same problem is solved a large number of time (Law of Large Number).
- In some cases the expectation is not really representative of your risk attitude. Lets consider two examples:
  - Are you ready to pay 1000€ to have one chance over ten to win 10000€ ?
  - You need to be at the airport in 1 hour or you miss your flight, you have the choice between two mean of transport, one of them take surely 50', the other take 40' four times out of five, and 70' one time out of five.

# Alternative cost functions II

Here are some cost functions you might consider

- Probability of reaching a given level of cost : $\mathbb{P}(L(u, \boldsymbol{\xi}) \leq 0)$
- Value-at-Risk of costs $V@R_\alpha(L(u, \boldsymbol{\xi}))$, where for any real valued random variable $\boldsymbol{X}$,

$$V@R_\alpha(\boldsymbol{X}) := \inf_{t \in \mathbb{R}} \left\{ \mathbb{P}(\boldsymbol{X} \geq t) \leq \alpha \right\}.$$

  In other word there is only a probability of $\alpha$ of obtaining a cost worse than $V@R_\alpha(\boldsymbol{X})$.
- Average Value-at-Risk of costs $AV@R_\alpha(L(u, \boldsymbol{\xi}))$, which is the expected cost over the $\alpha$ worst outcomes.

# Alternative constraints I

- The natural extension of the deterministic constraint $g(u, \xi) \leq 0$ to $g(u, \boldsymbol{\xi}) \leq 0 \; \mathbb{P} - as$ can be extremely conservative, and even often without any admissible solutions.
- For example, if $u$ is a level of production that need to be greater than the demand. In a deterministic setting the realized demand is equal to the forecast. In a stochastic setting we add an error to the forecast. If the error is unbouded (e.g. Gaussian) no control $u$ is admissible.

# Alternative constraints II

Here are a few possible constraints

- $\mathbb{E}[g(u, \boldsymbol{\xi})] \leq 0$, for quality of service like constraint.
- $\mathbb{P}(g(u, \boldsymbol{\xi}) \leq 0) \geq 1 - \alpha$ for chance constraint. Chance constraint is easy to present, but might lead to misconception as nothing is said on the event where the constraint is not satisfied.
- $AV@R_\alpha(g(u, \boldsymbol{\xi})) \leq 0$

# Presentation Outline

# Computing expectation

- Computing an expectation $\mathbb{E}\left[L(u, \xi)\right]$ for a given $u$ is costly.
- If $\xi$ is a r.v. with known law admitting a density, $\mathbb{E}\left[L(u, \xi)\right]$ is a (multidimensional) integral.
- If $\xi$ is a r.v. with known discrete law, $\mathbb{E}\left[L(u, \xi)\right]$ is a sum over all possible realizations of $\xi$, which can be huge.
- If $\xi$ is a r.v. that can be simulated but with unknown law, $\mathbb{E}\left[L(u, \xi)\right]$ cannot be computed exactly.

Solution : use Law of Large Number (LLN) and Central Limit Theorem (CLT).

- Draw $N \simeq 1000$ realization of $\xi$.
- Compute the sample average $\frac{1}{N}\sum_{i=1}^{N} L(u, \xi_i)$.
- Use CLT to give an asymptotic confidence interval of the expectation.

This is known as the Monte-Carlo method.

# Computing expectation

- Computing an expectation $\mathbb{E}\left[L(u, \xi)\right]$ for a given $u$ is costly.
- If $\xi$ is a r.v. with known law admitting a density, $\mathbb{E}\left[L(u, \xi)\right]$ is a (multidimensional) integral.
- If $\xi$ is a r.v. with known discrete law, $\mathbb{E}\left[L(u, \xi)\right]$ is a sum over all possible realizations of $\xi$, which can be huge.
- If $\xi$ is a r.v. that can be simulated but with unknown law, $\mathbb{E}\left[L(u, \xi)\right]$ cannot be computed exactly.

Solution : use Law of Large Number (LLN) and Central Limit Theorem (CLT).

- Draw $N \simeq 1000$ realization of $\xi$.
- Compute the sample average $\frac{1}{N} \sum_{i=1}^{N} L(u, \xi_i)$.
- Use CLT to give an asymptotic confidence interval of the expectation.

This is known as the Monte-Carlo method.

# Consequence : evaluating a solution is difficult

- In stochastic optimization even evaluating the value of a solution can be difficult an require approximate methods.
- The same holds true for checking admissibility of a candidate solution.
- It is even more difficult to obtain first order information (subgradient).

Standard solution : sampling and solving the sampled problem (Sample Average Approximation).

# Consequence : evaluating a solution is difficult

- In stochastic optimization even evaluating the value of a solution can be difficult an require approximate methods.
- The same holds true for checking admissibility of a candidate solution.
- It is even more difficult to obtain first order information (subgradient).

Standard solution : sampling and solving the sampled problem (Sample Average Approximation).

# Optimization problem and simulator

- Generally speaking stochastic optimization problem are not well posed and often need to be approximated before solving them.
- Good practice consists in defining a simulator, i.e. a representation of the "real problem" on which solution can be tested.
- Then find a candidate solution by solving an (or multiple) approximated problem.
- Finally evaluate the candidate solutions on the simulator. The comparison can be done on more than one dimension (e.g. constraints, risk...)

# Presentation Outline

# One-Stage Problems

Assume that $\boldsymbol{\xi}$ has a discrete distribution [1] , with $\mathbb{P}(\boldsymbol{\xi} = \xi^s) = \pi^s > 0$ for $s \in [\![1, S]\!]$. Then, the one-stage problem

$$\begin{aligned} \min_{u_0} \quad & \mathbb{E}\Big[L(u_0, \boldsymbol{\xi})\Big] \\ s.t. \quad & g(u_0, \boldsymbol{\xi}) \leq 0, \qquad \mathbb{P}-a.s \end{aligned}$$

can be written

$$\begin{aligned} \min_{u_0} \quad & \sum_{s=1}^{S} \pi^s L(u_0, \xi^s) \\ s.t \quad & g(u_0, \xi^s) \leq 0, \qquad \forall s \in [\![1, S]\!]. \end{aligned}$$

[1] If the distribution is continuous we can sample and work on the sampled distribution, this is called the Sample Average Approximation approach with lots of guarantee and results

# Newsvendor problem (continued)

We assume that the demand can take value $\{d^s\}_{s\in[\![1,S]\!]}$ with probabilities $\{\pi^s\}_{s\in[\![1,S]\!]}$.

# Newsvendor problem (continued)

We assume that the demand can take value $\{d^s\}_{s\in[\![1,S]\!]}$ with probabilities $\{\pi^s\}_{s\in[\![1,S]\!]}$.
In this case the stochastic newsvendor problem reads

$$\begin{aligned} \min_{u} \quad & \sum_{s=1}^{S} \pi^s \Big( cu - p \min(u, d^s) \Big) \\ s.t. \quad & u \geq 0 \end{aligned}$$

# Presentation Outline

# Recourse Variable

In most problem we can make a correction $u_1$ once the uncertainty is known:

$$u_0 \rightsquigarrow \xi_1 \rightsquigarrow u_1.$$

As recourse control $u_1$ is a function of $\xi$ it is a random variable, the two-stage optimization problem then reads

$$\begin{aligned} \min_{u_0, \boldsymbol{u}_1} \quad & \mathbb{E}\Big[L(u_0, \xi, \boldsymbol{u}_1)\Big] \\ s.t. \quad & g(u_0, \xi, \boldsymbol{u}_1) \leq 0, \qquad \mathbb{P}-a.s \\ & \boldsymbol{u}_1 \preceq \xi \end{aligned}$$

- $u_0$ is called a first stage control
- $u_1$ is called a second stage (or recourse) control

# Recourse Variable

In most problem we can make a correction $u_1$ once the uncertainty is known:

$$u_0 \rightsquigarrow \xi_1 \rightsquigarrow u_1.$$

As recourse control $u_1$ is a function of $\xi$ it is a random variable, the two-stage optimization problem then reads

$$\begin{array}{ll} \min\limits_{u_0, \boldsymbol{u}_1} & \mathbb{E}\Big[L(u_0, \xi, \boldsymbol{u}_1)\Big] \\ s.t. & g(u_0, \xi, \boldsymbol{u}_1) \leq 0, \qquad \mathbb{P}-a.s \\ & \boldsymbol{u}_1 \preceq \xi \end{array}$$

- $u_0$ is called a first stage control
- $u_1$ is called a second stage (or recourse) control

# Two-stage Problem

The extensive formulation of

$$\begin{aligned} \min_{u_0, \boldsymbol{u}_1} \quad & \mathbb{E}\Big[L(u_0, \boldsymbol{\xi}, \boldsymbol{u}_1)\Big] \\ s.t. \quad & g(u_0, \boldsymbol{\xi}, \boldsymbol{u}_1) \leq 0, \qquad \mathbb{P}-a.s \\ & \boldsymbol{u}_1 \preceq \boldsymbol{\xi} \end{aligned}$$

is

$$\begin{aligned} \min_{u_0, \{u_1^s\}_{s\in[\![1,S]\!]}} \quad & \sum_{s=1}^{S} \pi^s L(u_0, \xi^s, u_1^s) \\ s.t \quad & g(u_0, \xi^s, u_1^s) \leq 0, \qquad \forall s \in [\![1, S]\!]. \end{aligned}$$

It is a deterministic problem that can be solved with standard tools or specific methods.

# Two-stage newsvendor problem I

We can represent the newsvendor problem in a 2-stage framework.

- Let $u_0$ be the number of newspaper bought in the morning.
- let $u_1$ be the number of newspaper sold during the day.

# Two-stage newsvendor problem I

We can represent the newsvendor problem in a 2-stage framework.

- Let $u_0$ be the number of newspaper bought in the morning.
  $\rightsquigarrow$ first stage control
- let $u_1$ be the number of newspaper sold during the day.
  $\rightsquigarrow$ second stage control

# Two-stage newsvendor problem I

We can represent the newsvendor problem in a 2-stage framework.

- Let $u_0$ be the number of newspaper bought in the morning.
  $\rightsquigarrow$ first stage control
- let $u_1$ be the number of newspaper sold during the day.
  $\rightsquigarrow$ second stage control

The problem reads

$$\begin{aligned} \min_{u_0, \boldsymbol{u}_1} \quad & \mathbb{E}\Big[c u_0 - p \boldsymbol{u}_1\Big] \\ s.t. \quad & u_0 \geq 0 \\ & \boldsymbol{u}_1 \leq u_0 \qquad \mathbb{P}-as \\ & \boldsymbol{u}_1 \leq \boldsymbol{d} \qquad \mathbb{P}-as \\ & \sigma(\boldsymbol{u}_1) \subset \sigma(\boldsymbol{d}) \end{aligned}$$

# Two-stage newsvendor problem II

In extensive formulation the problem reads

$$\begin{aligned} \min_{u_0, \{u_1^s\}_{s\in[\![1,S]\!]}} \quad & \sum_{s=1}^{S} \pi^s \left(c u_0 - p u_1^s\right) \\ s.t. \quad & u_0 \geq 0 \\ & u_1^s \leq u_0 && \forall s \in [\![1, S]\!] \\ & u_1^s \leq d^s && \forall s \in [\![1, S]\!] \end{aligned}$$

Note that there are as many second-stage control $u_1^s$ as there are possible realization of the demand $\boldsymbol{d}$, but only one first-stage control $u_0$.

# Two-stage newsvendor problem II

In extensive formulation the problem reads

$$\begin{aligned} \min_{u_0, \{u_1^s\}_{s\in[\![1,S]\!]}} \quad & \sum_{s=1}^{S} \pi^s \left(c u_0 - p u_1^s\right) \\ s.t. \quad & u_0 \geq 0 \\ & u_1^s \leq u_0 && \forall s \in [\![1, S]\!] \\ & u_1^s \leq d^s && \forall s \in [\![1, S]\!] \end{aligned}$$

Note that there are as many second-stage control $u_1^s$ as there are possible realization of the demand $\boldsymbol{d}$, but only one first-stage control $u_0$.

# Presentation Outline

# Two-stage framework : three information models

Consider the problem

$$\min_{\boldsymbol{u}_0, \boldsymbol{u}_1} \mathbb{E}\left[L(\boldsymbol{u}_0, \boldsymbol{\xi}, \boldsymbol{u}_1\right]$$

- Open-Loop approach : $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$ are deterministic. In this case both controls are choosen without any knowledge of the alea $\boldsymbol{\xi}$. The set of control is small, and an optimal control can be found through specific method (e.g. Stochastic Gradient).
- Two-Stage approach : $\boldsymbol{u}_0$ is deterministic and $\boldsymbol{u}_1$ is measurable with respect to $\boldsymbol{\xi}$. This is the problem tackled by Stochastic Programming method.
- Anticipative approach : $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$ are measurable with respect to $\boldsymbol{\xi}$. This approach consists in solving one deterministic problem per possible outcome of the alea, and taking the expectation of the value of this problems.

# Comparing the models

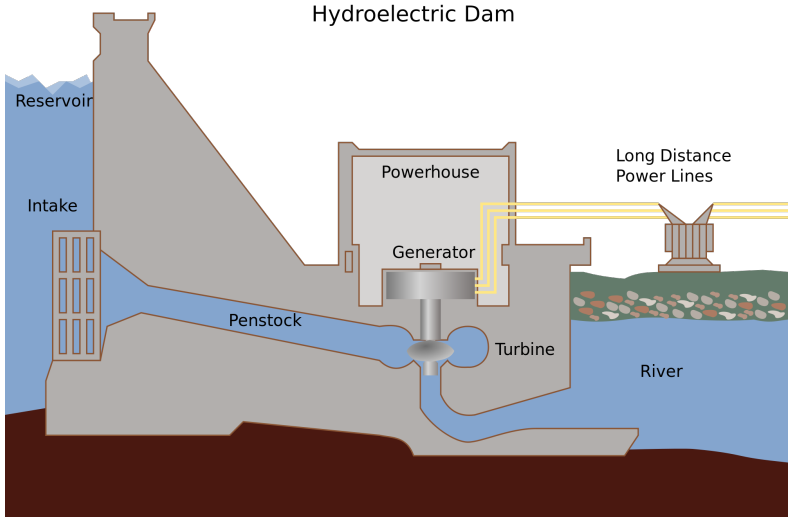
- By simple comparison of constraints we have

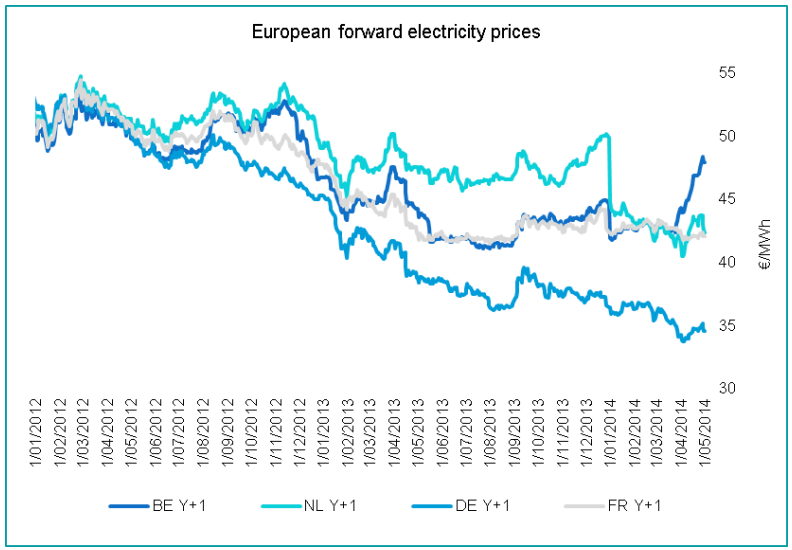
$$V^{anticipative} \leq V^{2-stage} \leq V^{OL}.$$

- $V^{OL}$ can be approximated through specific methods (e.g. Stochastic Gradient).
- $V^{2-stage}$ is obtained through Stochastic Programming specific methods. There are two main approaches:
  - Lagrangian decomposition methods (like Progressive-Hedging algorithm).
  - Benders decomposition methods (like L-shaped or nested-decomposition methods).
- $V^{anticipative}$ is difficult to compute exactly but can be estimated through Monte-Carlo approach by drawing a reasonable number of realizations of $\boldsymbol{\xi}$, solving the deterministic problem for each realization $\xi_i$ and taking the means of the value of the deterministic problem.

# Presentation Outline

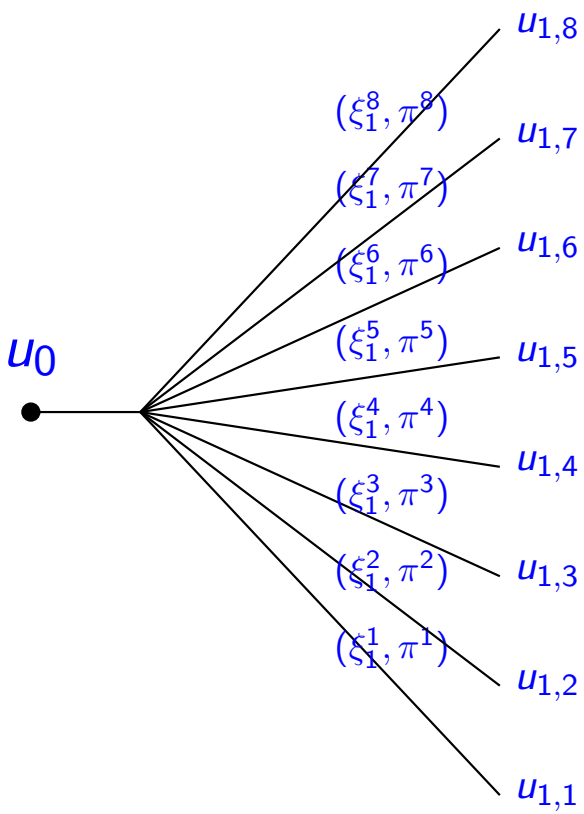
# Managing a dam




A dam can be seen as a battery, with random inflow of free electricity to be used at the best time.

# Where do we come from: two-stage programming




- We take decisions in two stages

$$u_0 \rightsquigarrow \boldsymbol{\xi}_1 \rightsquigarrow \boldsymbol{u}_1 \ ,$$

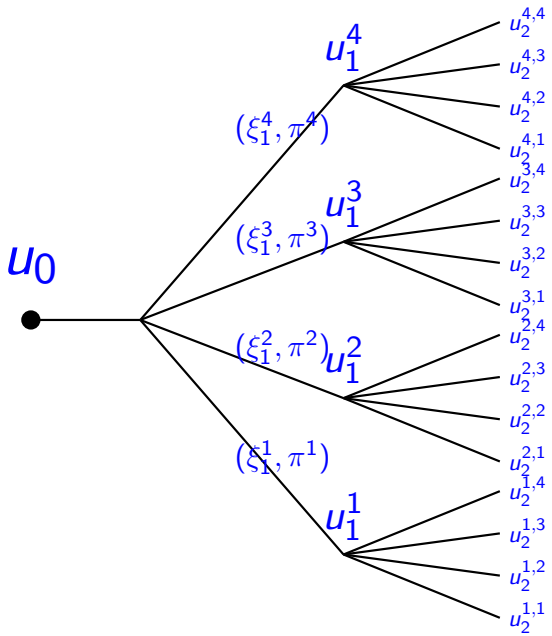
with $\boldsymbol{u}_1$: recourse decision .

- On a tree, it means solving the extensive formulation:

$$\min_{u_0, u_{1,s}} \sum_{s \in \mathbb{S}} \pi_s \left[ \langle c_s \ , u_0 \rangle + \langle p_s \ , u_{1,s} \rangle \right] \ .$$

We have as many $u_{1,s}$ as scenarios!

# Extending two-stage to multistage programming




$$\min_{\boldsymbol{u}} \ \mathbb{E}\big(j(\boldsymbol{u}, \boldsymbol{\xi})\big)$$

$$\boldsymbol{U} = (\boldsymbol{u}_0, \cdots, \boldsymbol{U}_T)$$

$$\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_T)$$

We take decisions in $T$ stages

$$\boldsymbol{\xi}_0 \rightsquigarrow \boldsymbol{u}_0 \rightsquigarrow \boldsymbol{\xi}_1 \rightsquigarrow \boldsymbol{u}_1 \rightsquigarrow \cdots \rightsquigarrow \boldsymbol{\xi}_T \rightsquigarrow \boldsymbol{u}_T \ .$$

# Introducing the non-anticipativity constraint

*We do not know what holds behind the door.*

## Non-anticipativity

At time $t$, decisions are taken sequentially, only knowing the past realizations of the perturbations.

Mathematically, this is equivalent to say that at time $t$, the decision $\boldsymbol{u}_t$ is

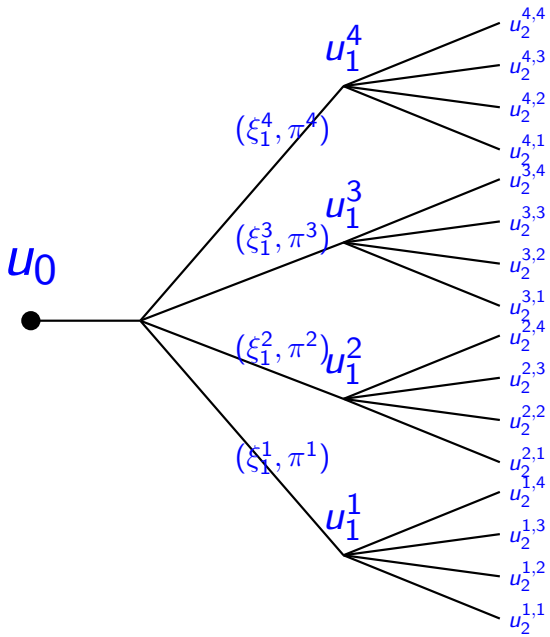
1. a function of past noises

$$\boldsymbol{u}_t = \pi_t(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_t) \,,$$

2. taken knowing the available information,

$$\sigma(\boldsymbol{u}_t) \subset \sigma(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_t) \,.$$

# Multistage extensive formulation approach




Assume that $\xi_t \in \mathbb{R}^{n_\xi}$ can take $n_\xi$ values and that $U_t(x)$ can take $n_u$ values.

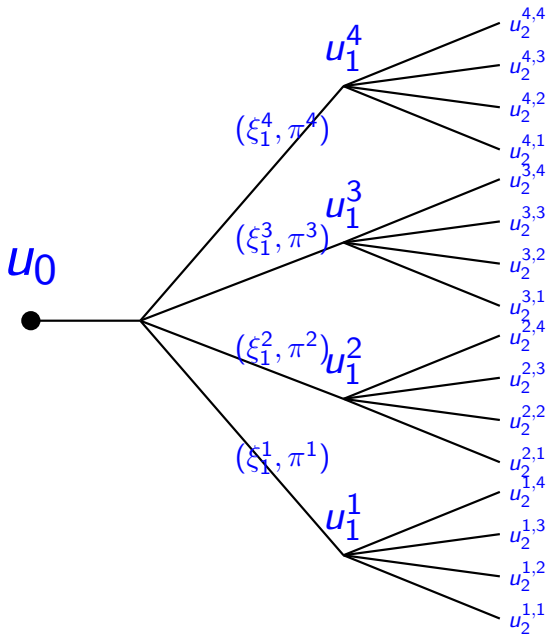
Then, considering the extensive formulation approach, we have

- $n_\xi^T$ scenarios.
- $(n_\xi^{T+1} - 1)/(n_\xi - 1)$ nodes in the tree.
- Number of variables in the optimization problem is roughly $n_u \times (n_\xi^{T+1} - 1)/(n_\xi - 1) \approx n_u n_\xi^T$.

The complexity grows exponentially with the number of stage. :-(

A way to overcome this issue is to compress information!

# Multistage extensive formulation approach




Assume that $\xi_t \in \mathbb{R}^{n_\xi}$ can take $n_\xi$ values and that $U_t(x)$ can take $n_u$ values.

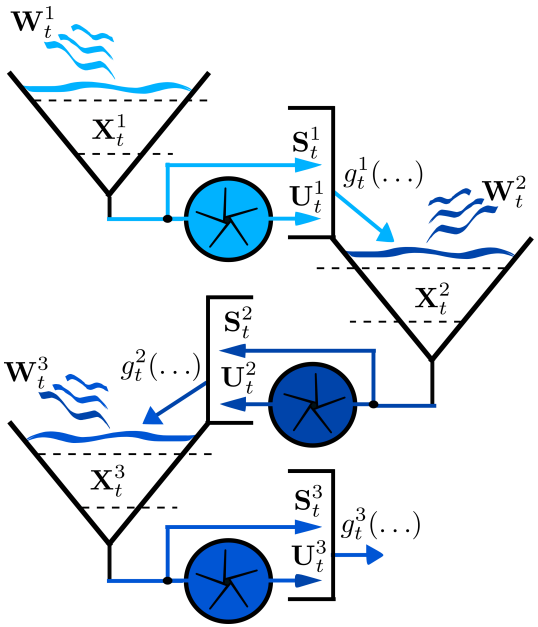
Then, considering the extensive formulation approach, we have

- $n_\xi^T$ scenarios.
- $(n_\xi^{T+1} - 1)/(n_\xi - 1)$ nodes in the tree.
- Number of variables in the optimization problem is roughly $n_u \times (n_\xi^{T+1} - 1)/(n_\xi - 1) \approx n_u n_\xi^T$.

The complexity grows exponentially with the number of stage. :-(

A way to overcome this issue is to compress information!

# Illustrating extensive formulation with the damsvalley example




- 5 interconnected dams
- 5 controls per timesteps
- 52 timesteps (one per week, over one year)
- $n_\xi = 10$ noises for each timestep

We obtain $10^{52}$ scenarios, and $\approx 5.10^{52}$ constraints in the extensive formulation ... Estimated storage capacity of the Internet: $10^{24}$ bytes.

# Presentation Outline

# Stochastic Controlled Dynamic System

A stochastic controlled dynamic system is defined by its dynamic

$$x_{t+1} = f_t(x_t, u_t, \xi_{t+1})$$

and initial state

$$x_0 = x_0$$

The variables

- $x_t$ is the state of the system,
- $u_t$ is the control applied to the system at time $t$,
- $\xi_t$ is an exogeneous noise.

# Examples

- Stock of water in a dam:
  - $x_t$ is the amount of water in the dam at time $t$,
  - $u_t$ is the amount of water turbined at time $t$,
  - $\xi_t$ is the inflow of water at time $t$.
- Boat in the ocean:
  - $x_t$ is the position of the boat at time $t$,
  - $u_t$ is the direction and speed chosen at time $t$,
  - $\xi_t$ is the wind and current at time $t$.
- Subway network:
  - $x_t$ is the position and speed of each train at time $t$,
  - $u_t$ is the acceleration chosen at time $t$,
  - $\xi_t$ is the delay due to passengers and incident on the network at time $t$.

# Optimization Problem

We want to solve the following optimization problem

$$\min \quad \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{x}_T)\Big] \tag{1a}$$

$$s.t. \quad \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_0 = x_0 \tag{1b}$$

$$\boldsymbol{u}_t \in U_t(\boldsymbol{x}_t) \tag{1c}$$

$$\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t := \sigma(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_t) \tag{1d}$$

Where

- constraint (1b) is the dynamic of the system ;
- constraint (1c) refer to the constraint on the controls;
- constraint (1d) is the information constraint : $\boldsymbol{u}_t$ is choosen knowing the realisation of the noises $\boldsymbol{\xi}_0, \ldots, \boldsymbol{\xi}_t$ but without knowing the realisation of the noises $\boldsymbol{\xi}_{t+1}, \ldots, \boldsymbol{\xi}_{T-1}$.

# Dynamic Programming Principle

## Theorem

*Assume that the noises $\boldsymbol{\xi}_t$ are independent and exogeneous. Then, there exists an optimal solution, called a strategy, of the form $\boldsymbol{u}_t = \pi_t(\boldsymbol{x}_t)$.*

*We have*

$$\pi_t(x) \in \underset{u \in U_t(x)}{\arg\min}\, \mathbb{E}\Big[ \underbrace{L_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1} \circ f_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{future costs}} \Big] ,$$

*where (Dynamic Programming Equation)*

$$\begin{cases} V_T(x) & = K(x) \\ V_t(x) & = \min\limits_{u \in U_t(x)} \mathbb{E}\Big[ L_t(x, u, \boldsymbol{\xi}_{t+1}) + V_{t+1} \circ \underbrace{f_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{"}\boldsymbol{x}_{t+1}\text{"}} \Big] \end{cases}$$

# Dynamic Programming Principle

## Theorem

*Assume that the noises $\boldsymbol{\xi}_t$ are independent and exogeneous. Then, there exists an optimal solution, called a strategy, of the form $\boldsymbol{u}_t = \pi_t(\boldsymbol{x}_t)$.*
*We have*

$$\pi_t(x) \in \underset{u \in U_t(x)}{\arg\min}\, \mathbb{E}\Big[ \underbrace{L_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{current cost}} + \underbrace{V_{t+1} \circ f_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{future costs}} \Big] ,$$

*where (Dynamic Programming Equation)*

$$\begin{cases} V_T(x) & = K(x) \\ V_t(x) & = \min\limits_{u \in U_t(x)} \mathbb{E}\Big[ L_t(x, u, \boldsymbol{\xi}_{t+1}) + V_{t+1} \circ \underbrace{f_t(x, u, \boldsymbol{\xi}_{t+1})}_{\text{"}\boldsymbol{x}_{t+1}\text{"}} \Big] \end{cases}$$

# Interpretation of Bellman Value

The Bellman's value function $V_{t_0}(x)$ can be interpreted as the value of the problem starting at time $t_0$ from the state $x$. More precisely we have

$$
\begin{aligned}
V_{t_0}(x) = \min \quad & \mathbb{E}\Big[\sum_{t=t_0}^{T-1} L_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}) + K(\boldsymbol{x}_T)\Big] \\
s.t. \quad & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\xi}_{t+1}), \qquad \boldsymbol{x}_{t_0} = x \\
& \boldsymbol{u}_t \in U_t(\boldsymbol{x}_t) \\
& \sigma(\boldsymbol{u}_t) \subset \sigma(\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_t)
\end{aligned}
$$

# Information structure I

In Problem (1), constraint (1d) is the information constraint.
There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

# Information structure I

In Problem (1), constraint (1d) is the information constraint.
There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

# Information structure I

In Problem (1), constraint (1d) is the information constraint.
There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

# Information structure I

In Problem (1), constraint (1d) is the information constraint.
There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

# Information structure I

In Problem (1), constraint (1d) is the information constraint.
There are different possible information structure.

- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_0$, the problem is open-loop, as the controls are choosen without knowledge of the realisation of any noise.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_t$, the problem is said to be in decision-hazard structure as decision $\boldsymbol{u}_t$ is chosen without knowing $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{t+1}$, the problem is said to be in hazard-decision structure as decision $\boldsymbol{u}_t$ is chosen with knowledge of $\boldsymbol{\xi}_{t+1}$.
- If constraint (1d) reads $\sigma(\boldsymbol{u}_t) \subset \mathcal{F}_{T-1}$, the problem is said to be anticipative as decision $\boldsymbol{u}_t$ is chosen with knowledge of all the noises.

# Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

# Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

# Information structure II

Be careful when modeling your information structure:

- Open-loop information structure might happen in practice (you have to decide on a planning and stick to it). If the problem does not require an open-loop solution then it might be largely suboptimal (imagine driving a car eyes closed...). In any case it yields an upper-bound of the problem.
- In some cases decision-hazard and hazard-decision are both approximation of the reality. Hazard-decision yield a lower value then decision-hazard.
- Anticipative structure is never an accurate modelization of the reality. However it can yield a lower-bound of your optimization problem relying on deterministic optimization and Monte-Carlo.

# Non-independence of noise in DP

- The Dynamic Programming equation requires only the time-independence of noises.
- This can be relaxed if we consider an extended state.
- Consider a dynamic system driven by an equation

$$y_{t+1} = f_t(x_t, u_t, \varepsilon_{t+1})$$

where the random noise $\varepsilon_t$ is an AR1 process :
$\varepsilon_t = \alpha_t \varepsilon_{t-1} + \beta_t + \xi_t$, where $\{\xi_t\}_{t \in \mathbb{Z}}$ are independent.

- Then $y_t$ is called the physical state of the system and DP can be used with the information state $x_t = (y_t, \varepsilon_{t-1})$.
- Generically speaking, if the noise $\xi_t$ is exogeneous (not affected by decisions $u_t$), then we can always apply Dynamic Programming with the state

$$(x_t, \xi_1, \ldots, \xi_t).$$

# Presentation Outline

**Algorithm 1:** Classical stochastic DP algorithm

**Data:** Problem parameters

**Result:** optimal strategy and value;

$V_T \equiv K$ ; $V_t \equiv 0$

**for** $t : T-1 \to 0$ **do**

  **for** $x \in \mathbb{X}_t$ **do**

$$V_t(x) = \min_{u \in \mathcal{U}_t(x)} \mathbb{E}\Big[ L_t(x, u, \boldsymbol{\xi}_{t+1}) + V_{t+1}\Big( \underbrace{f_t(x, u, \xi_{t+1})}_{x_{t+1}} \Big) \Big]$$

**Algorithm 1:** Classical stochastic DP algorithm

**Data:** Problem parameters

**Result:** optimal strategy and value;

$V_T \equiv K$ ; $V_t \equiv 0$

**for** $t : T-1 \to 0$ **do**

  **for** $x \in \mathbb{X}_t$ **do**

    $V_t(x) = +\infty$;

    **for** $u \in \mathcal{U}(x)$ **do**

      $$Q_t(x,u) = \mathbb{E}\Big[L_t(x,u,\boldsymbol{\xi}_{t+1}) + V_{t+1}\Big(\underbrace{f_t(x,u,\xi_{t+1})}_{x_{t+1}}\Big)\Big]$$

      **if** $Q_t(x,u) < V_t(x)$ **then**

        $V_t(x) = Q_t(x,u)$;

        $\pi_t(x) = u$;

**Algorithm 1:** Classical stochastic DP algorithm

**Data:** Problem parameters

**Result:** optimal strategy and value;

$V_T \equiv K$ ; $V_t \equiv 0$

**for** $t : T-1 \to 0$ **do**

  **for** $x \in \mathbb{X}_t$ **do**

    $V_t(x) = +\infty$;

    **for** $u \in \mathcal{U}(x)$ **do**

      **for** $\xi \in \Xi_{t+1}$ **do**

        $x_{t+1}^{\xi} = f_t(x, u, \xi)$;

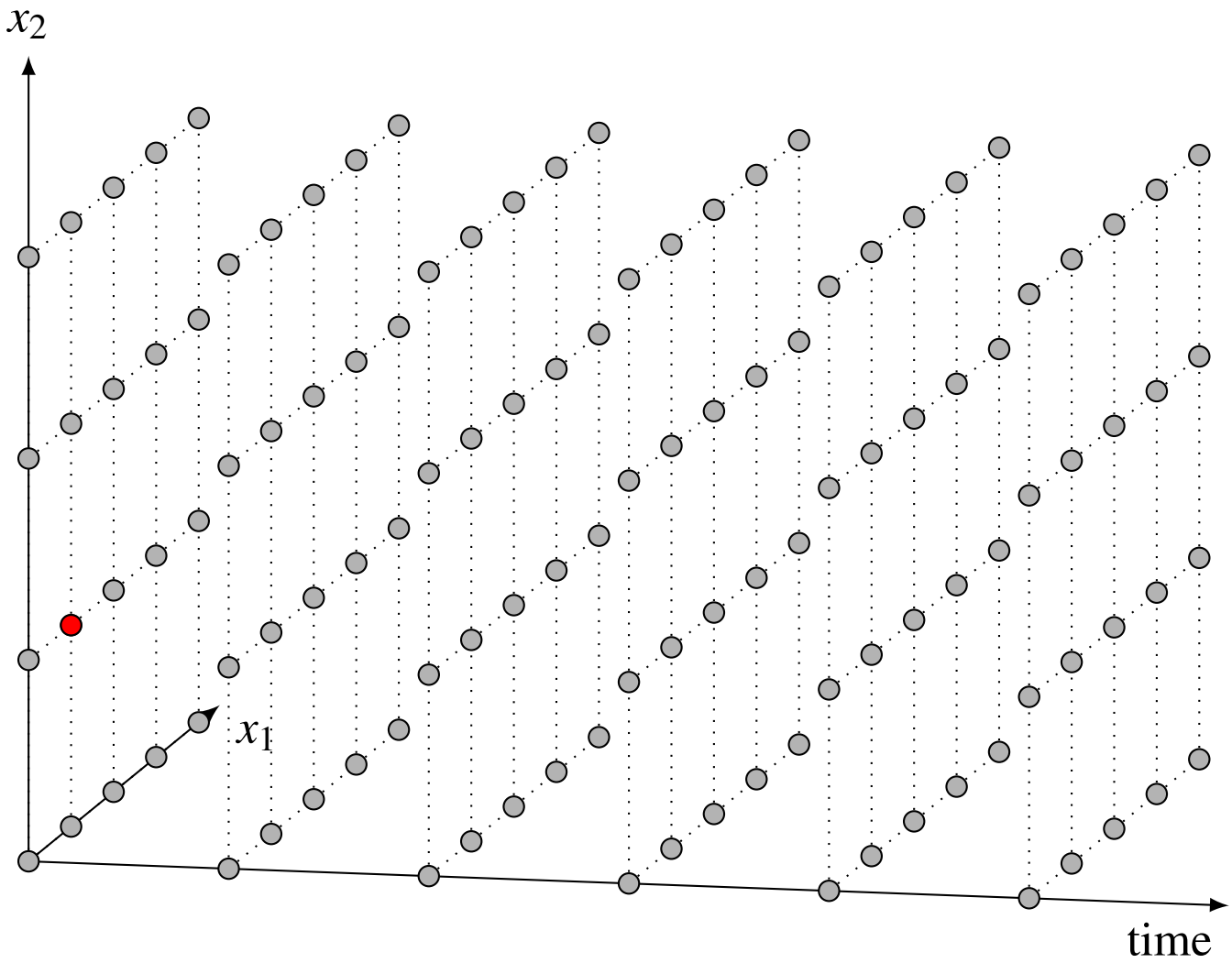
        $\dot{Q}_t(x, u, \xi) = L_t(x, u, \boldsymbol{\xi}_{t+1}) + V_{t+1}(x_{t+1}^{\xi}) + \mathbb{I}_{x_{t+1}^{\xi} \in X_t}$

      $Q_t(x, u) = \sum_{\xi \in \Xi_{t+1}} \mathbb{P}(\boldsymbol{\xi}_{t+1} = \xi) \dot{Q}_t(x, u, \xi)$;

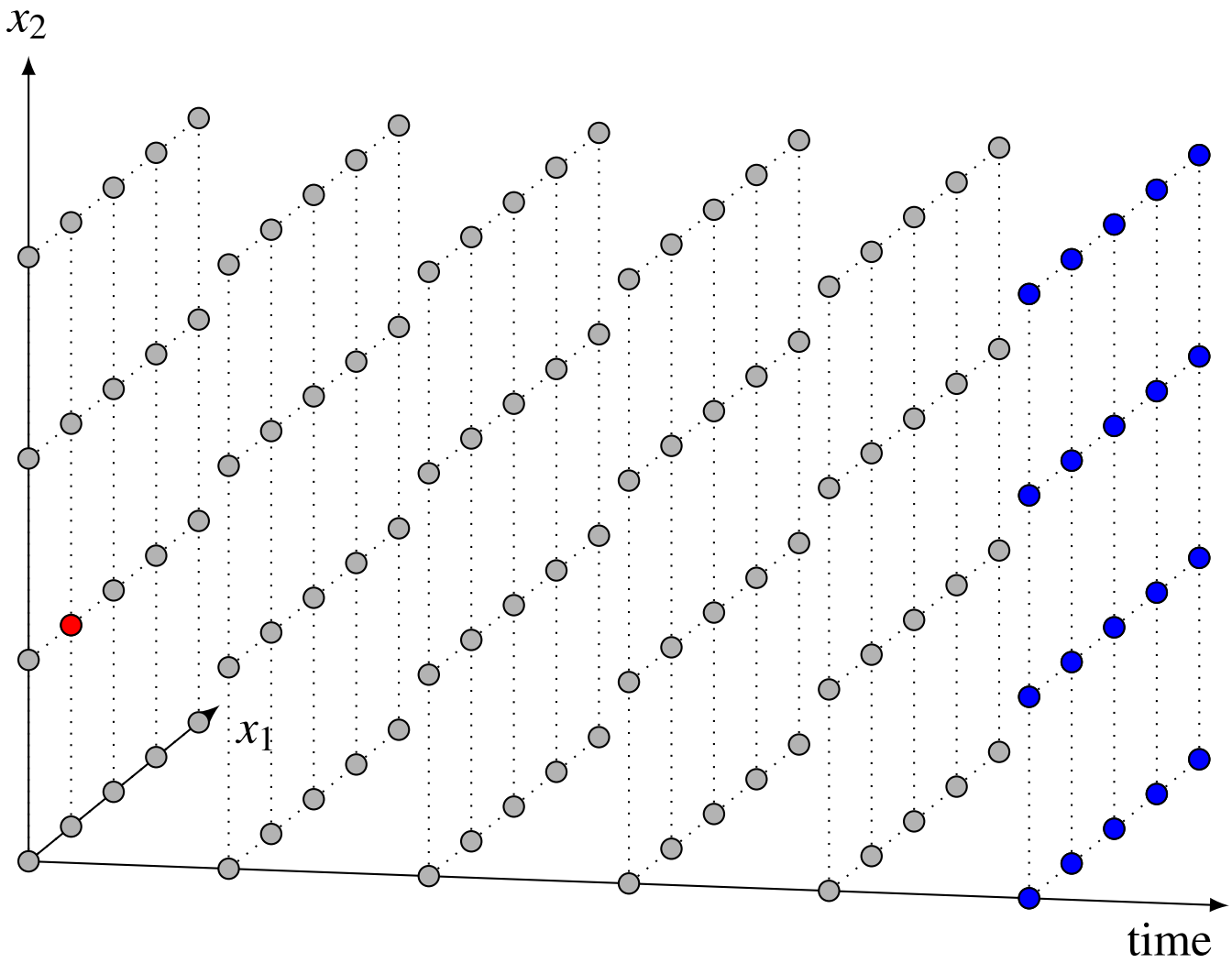
      **if** $Q_t(x, u) < V_t(x)$ **then**

        $V_t(x) = Q_t(x, u)$; $\quad \pi_t(x) = u$;

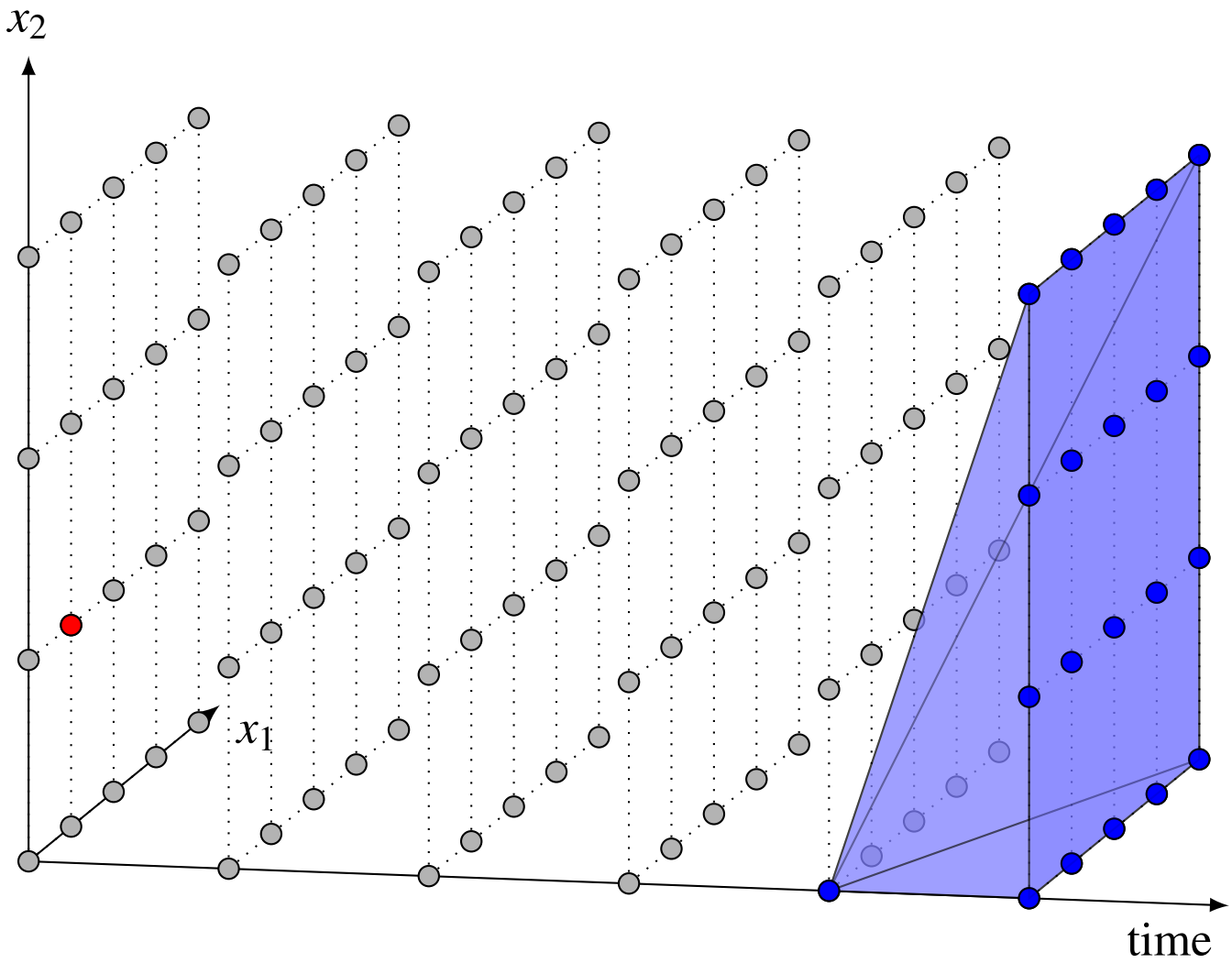
# Dynamic Programming: Illustration



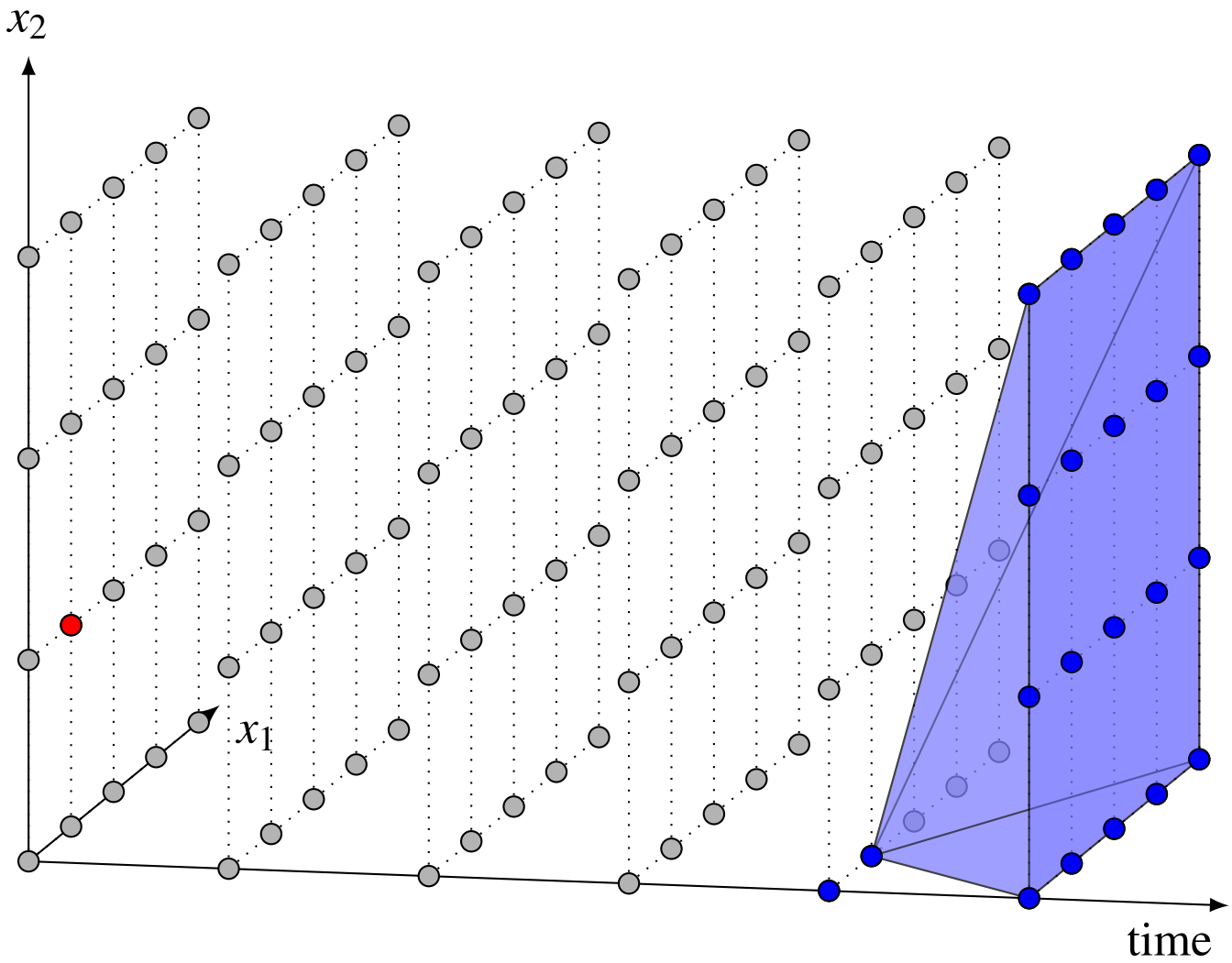
# Dynamic Programming: Illustration



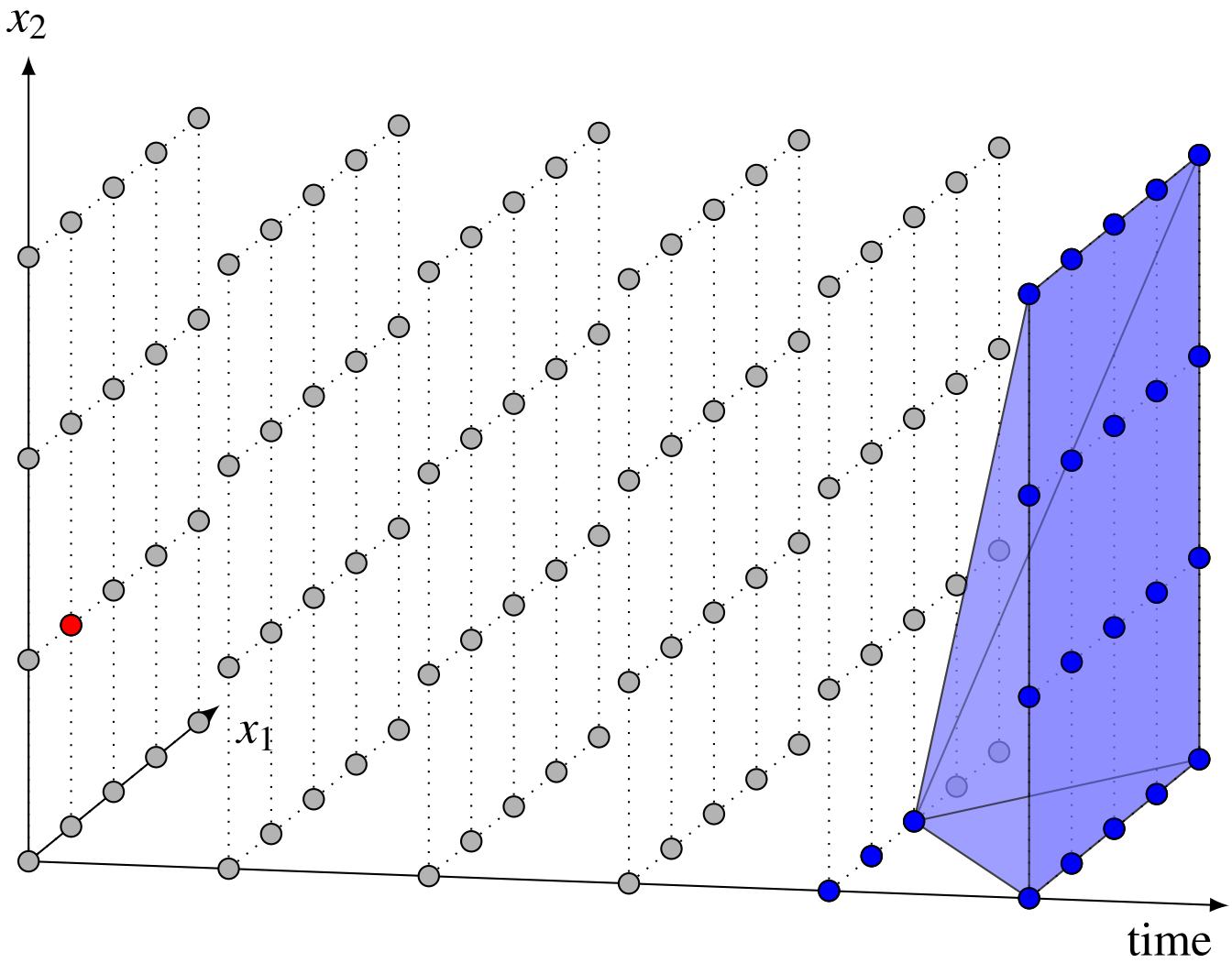
# Dynamic Programming: Illustration



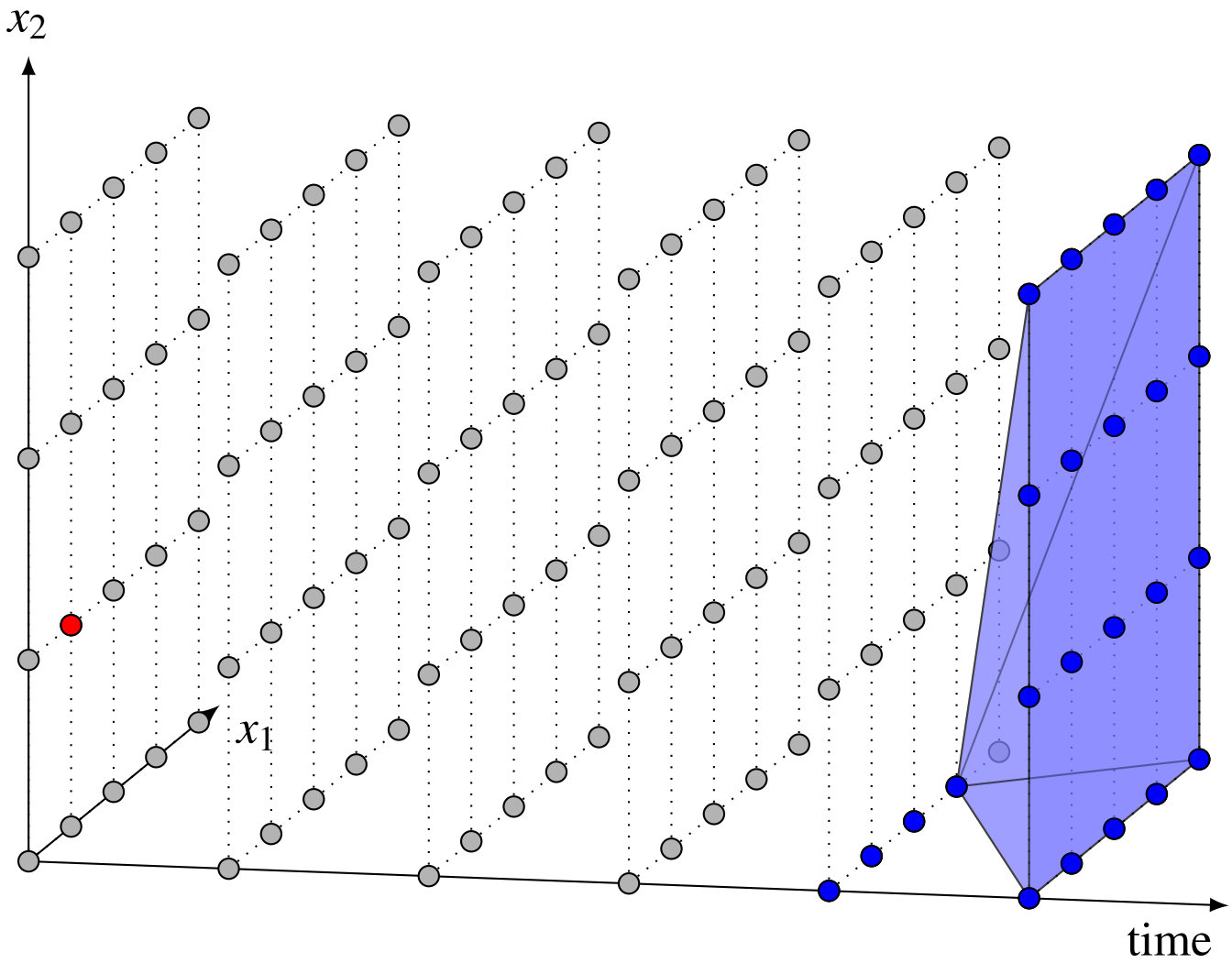
# Dynamic Programming: Illustration



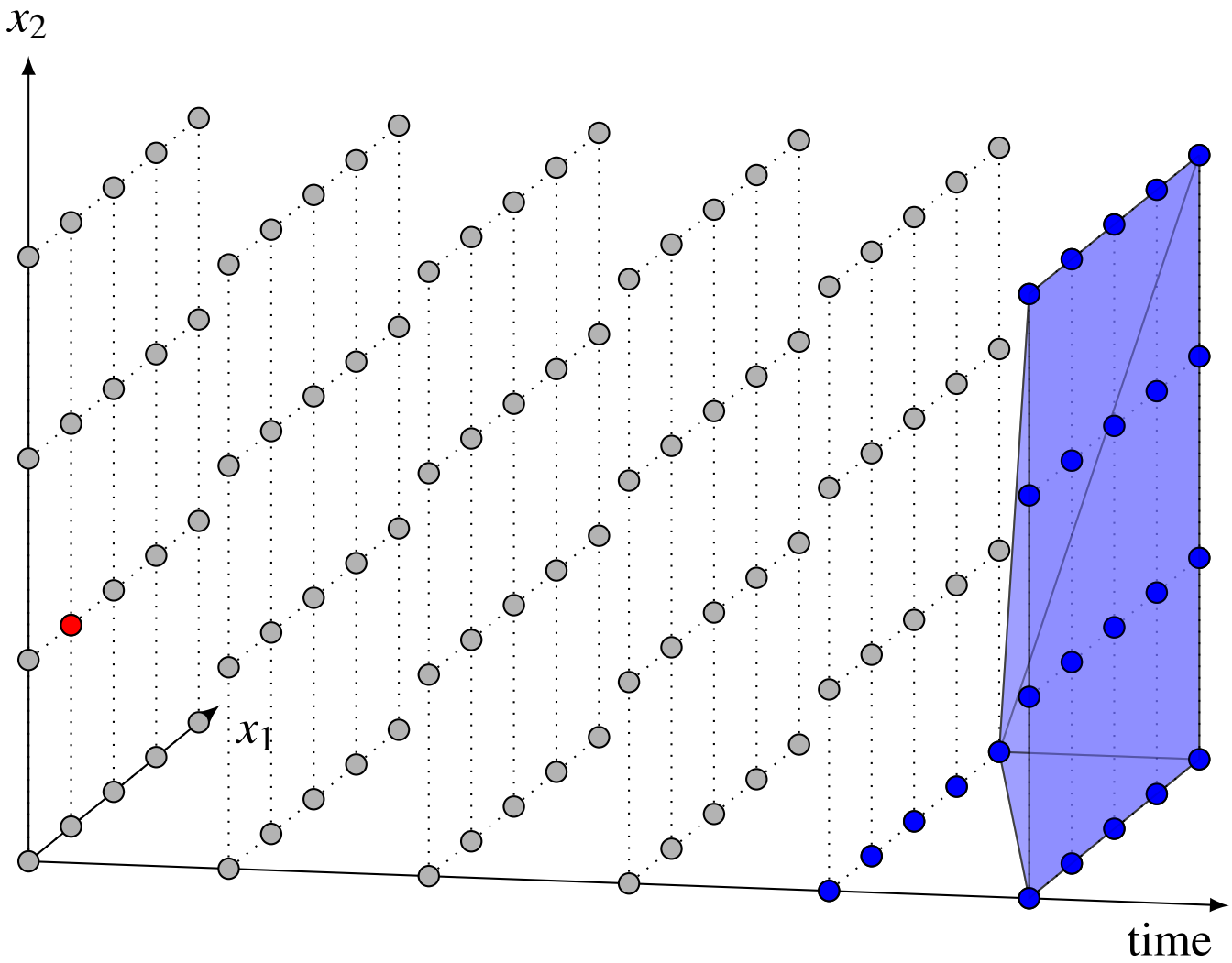
# Dynamic Programming: Illustration



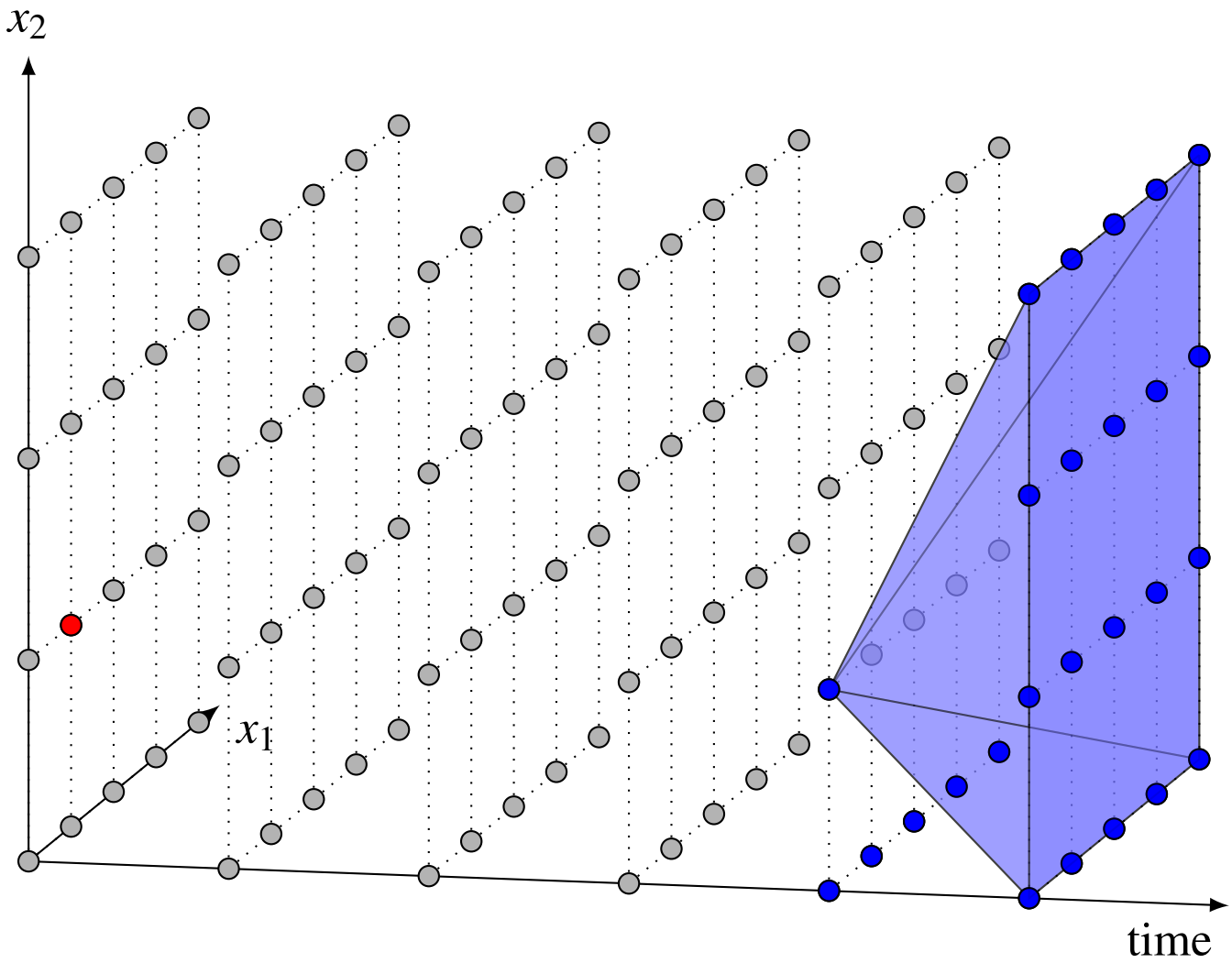
# Dynamic Programming: Illustration



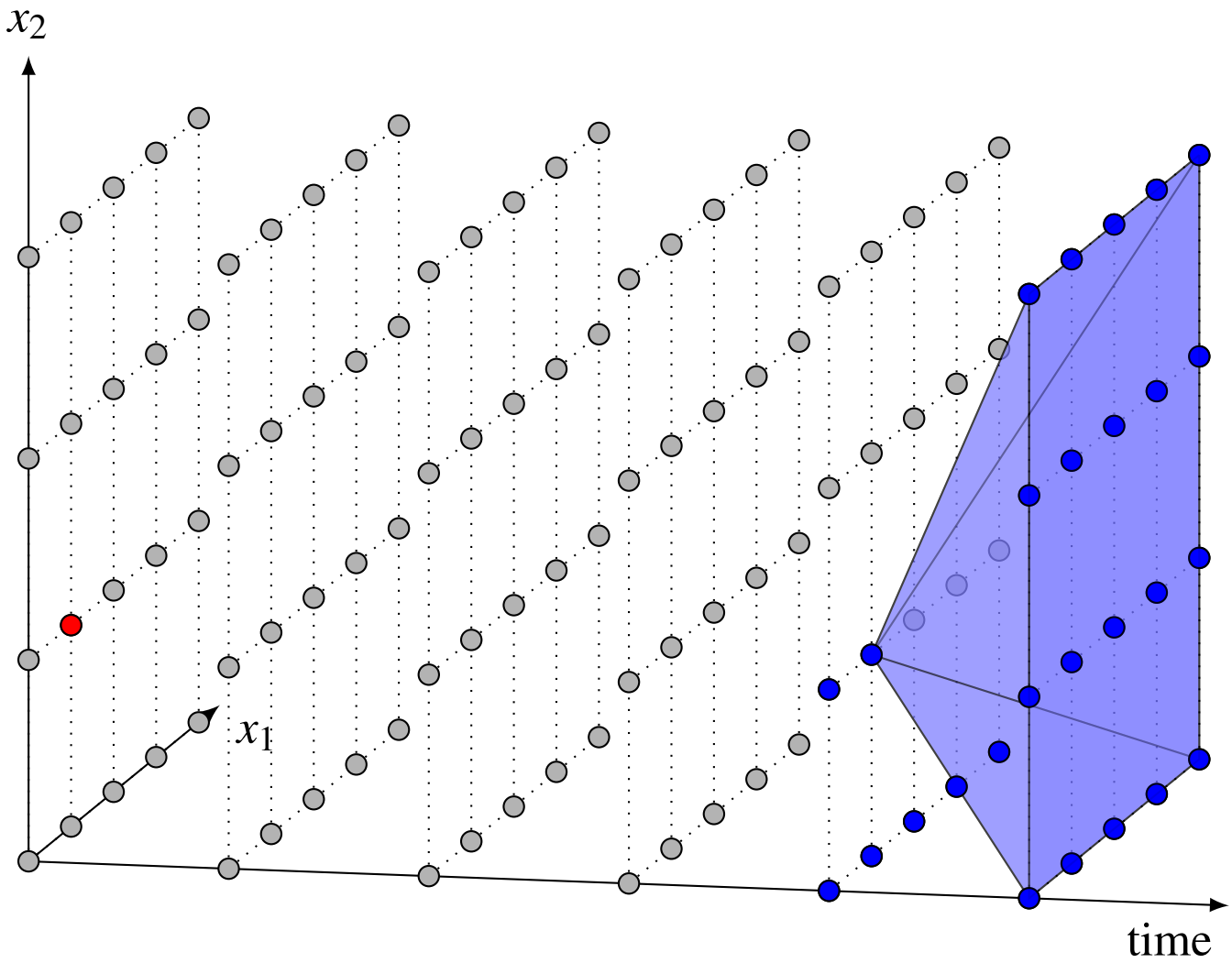
# Dynamic Programming: Illustration



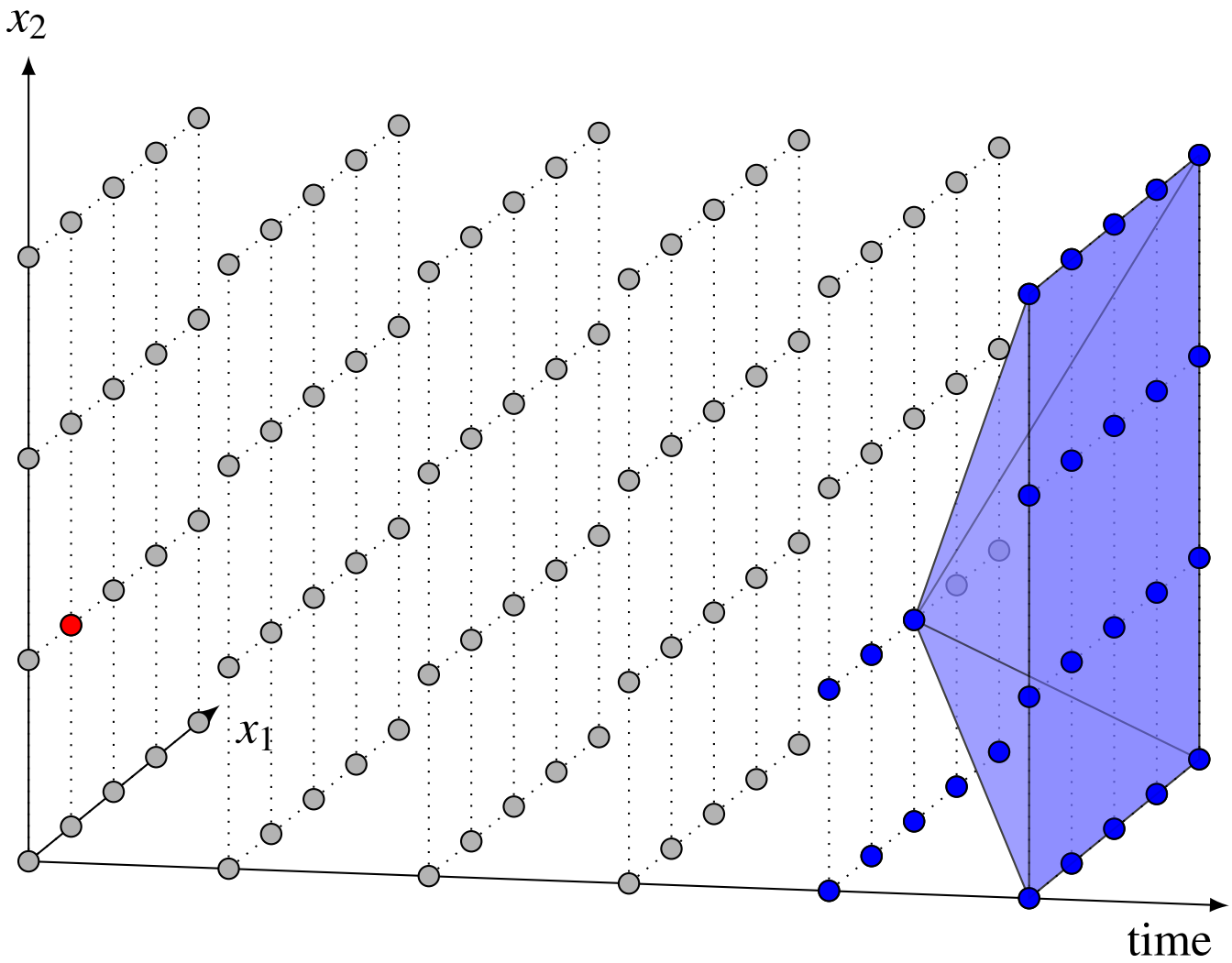
# Dynamic Programming: Illustration



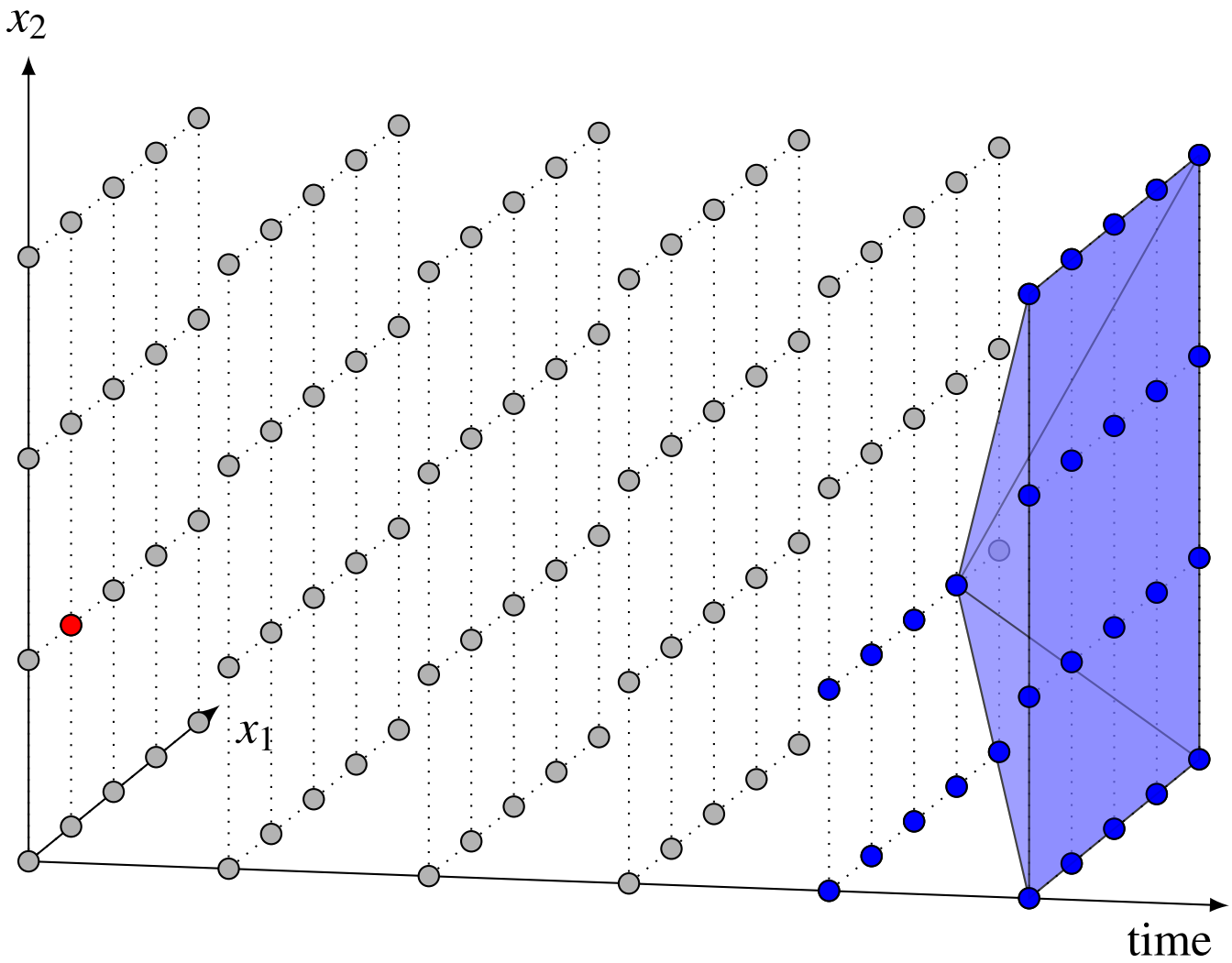
# Dynamic Programming: Illustration



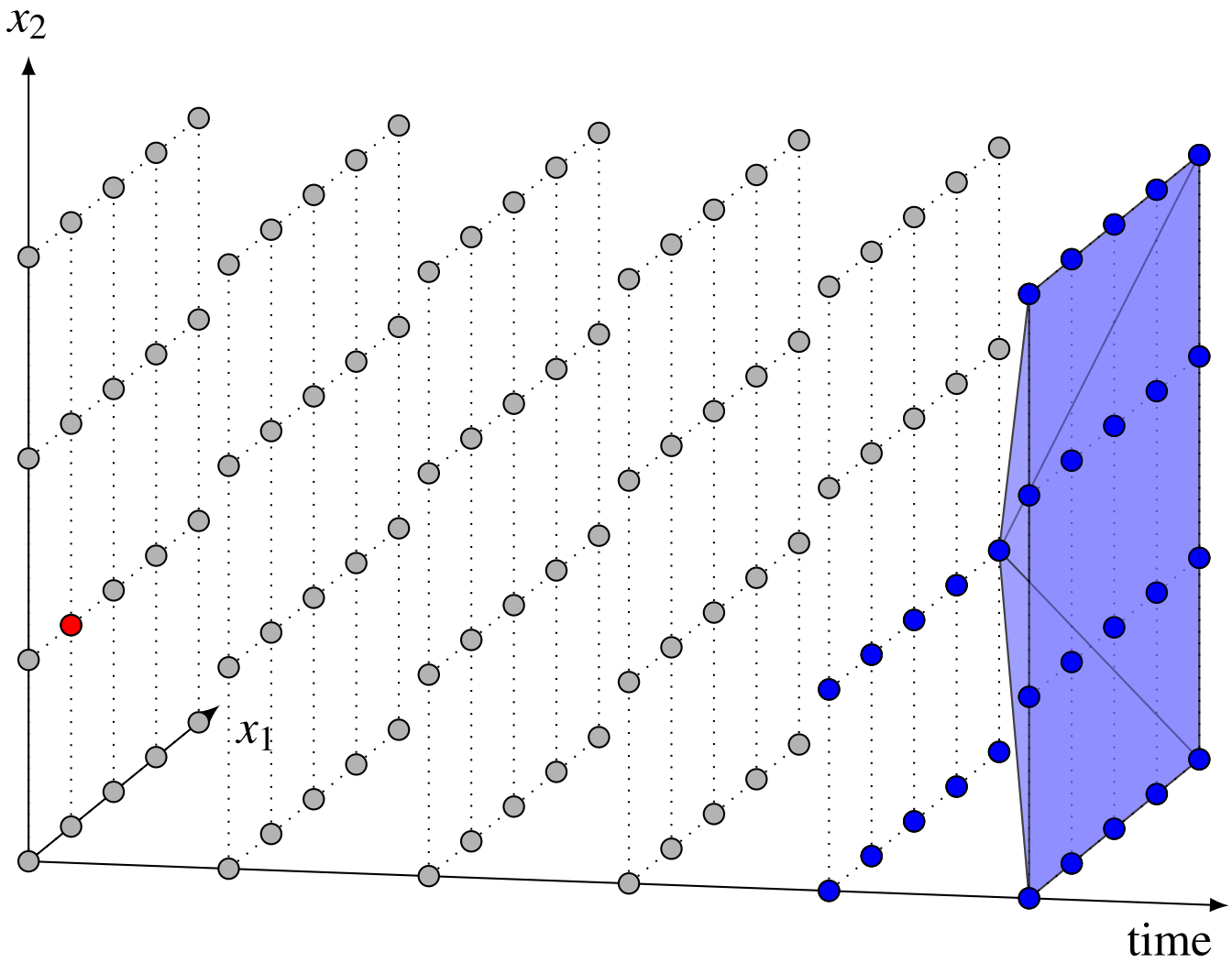
# Dynamic Programming: Illustration



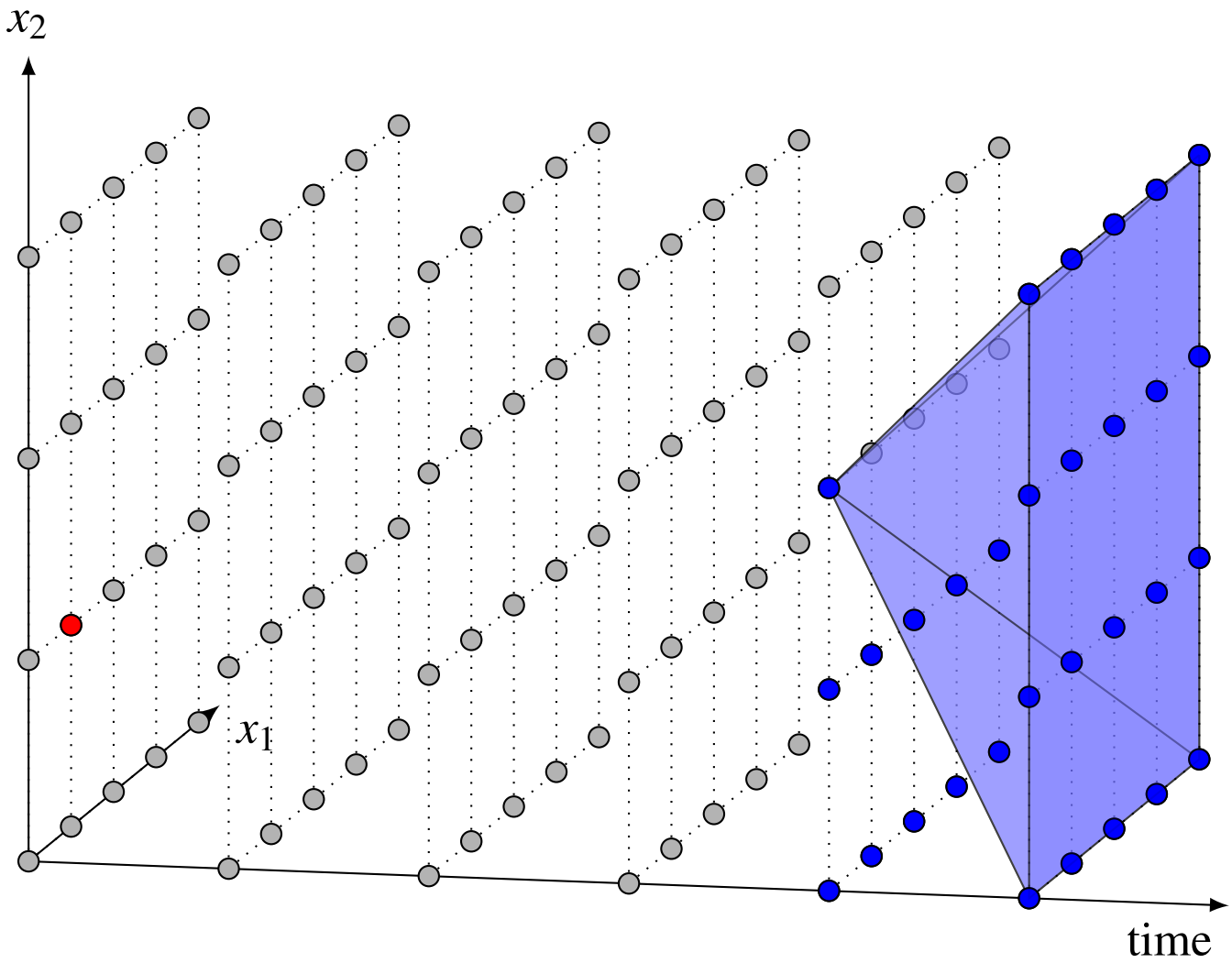
# Dynamic Programming: Illustration



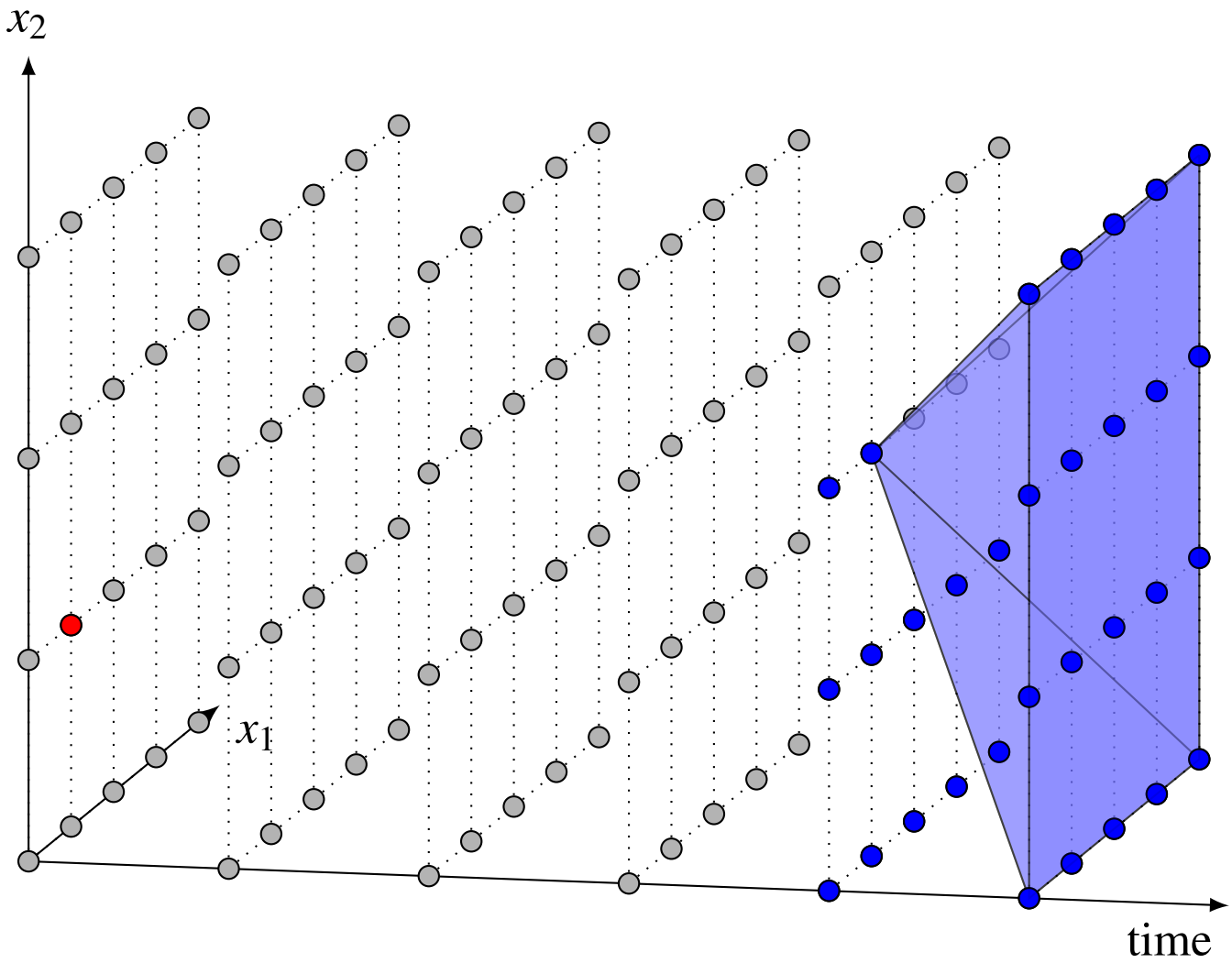
# Dynamic Programming: Illustration



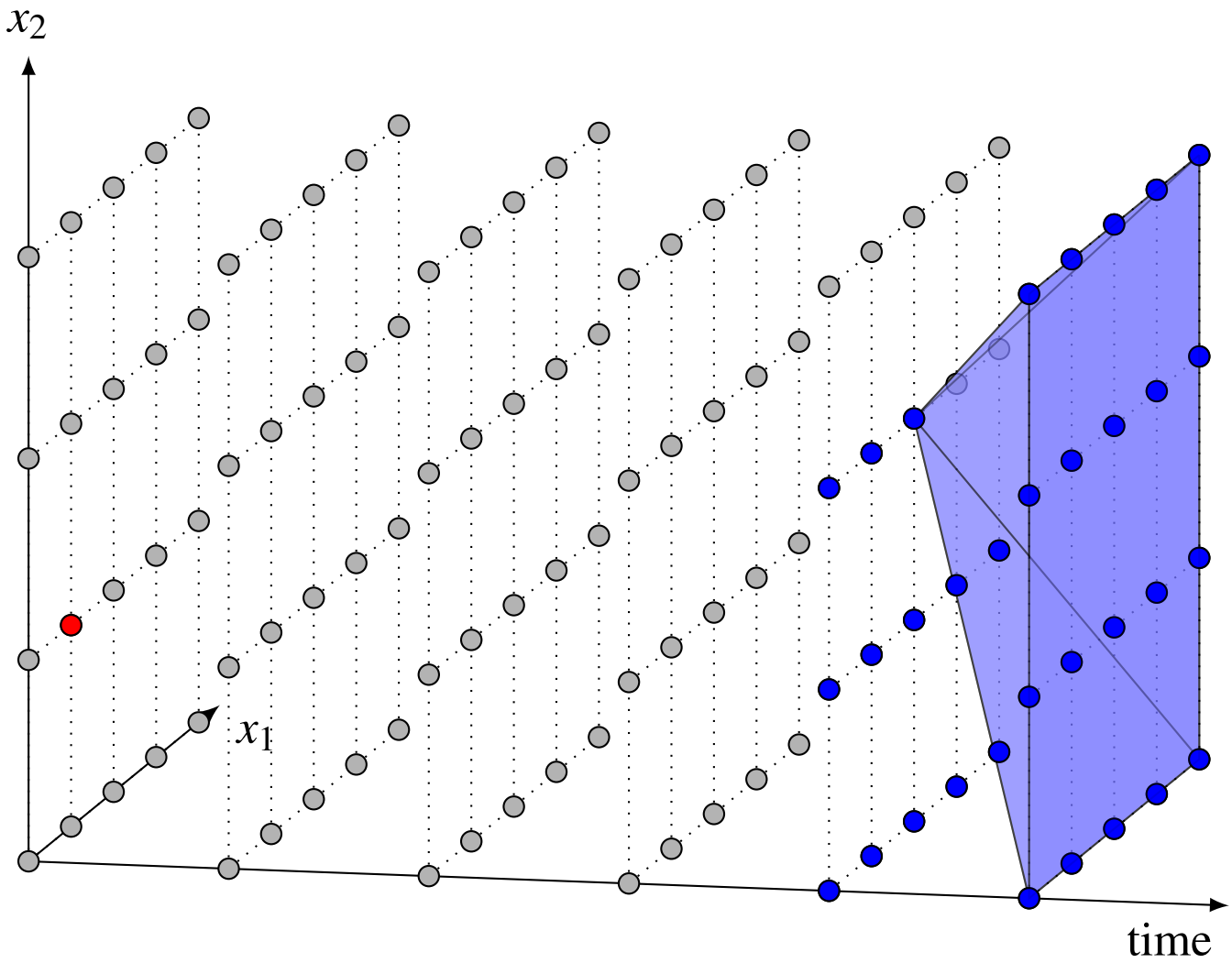
# Dynamic Programming: Illustration



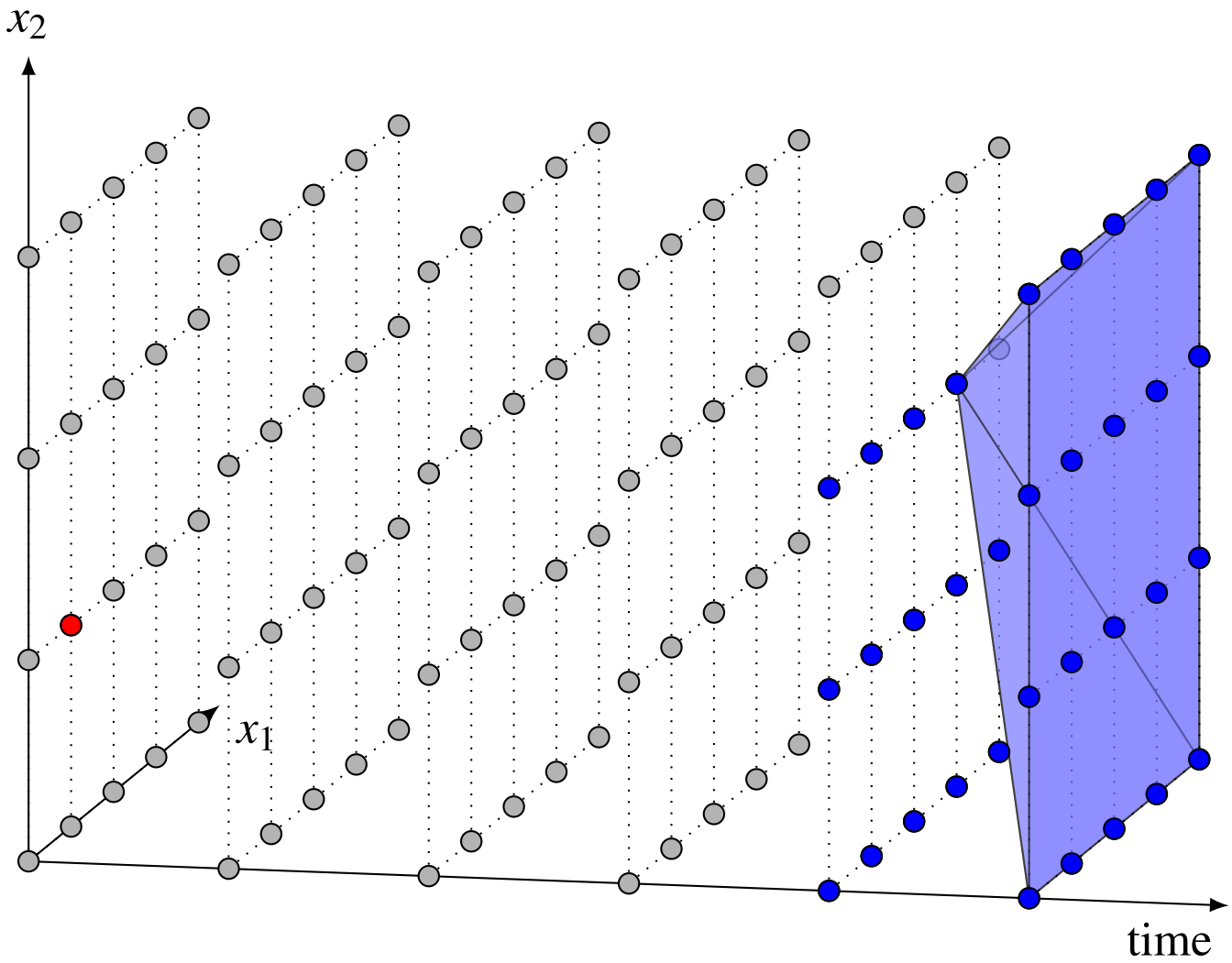
# Dynamic Programming: Illustration



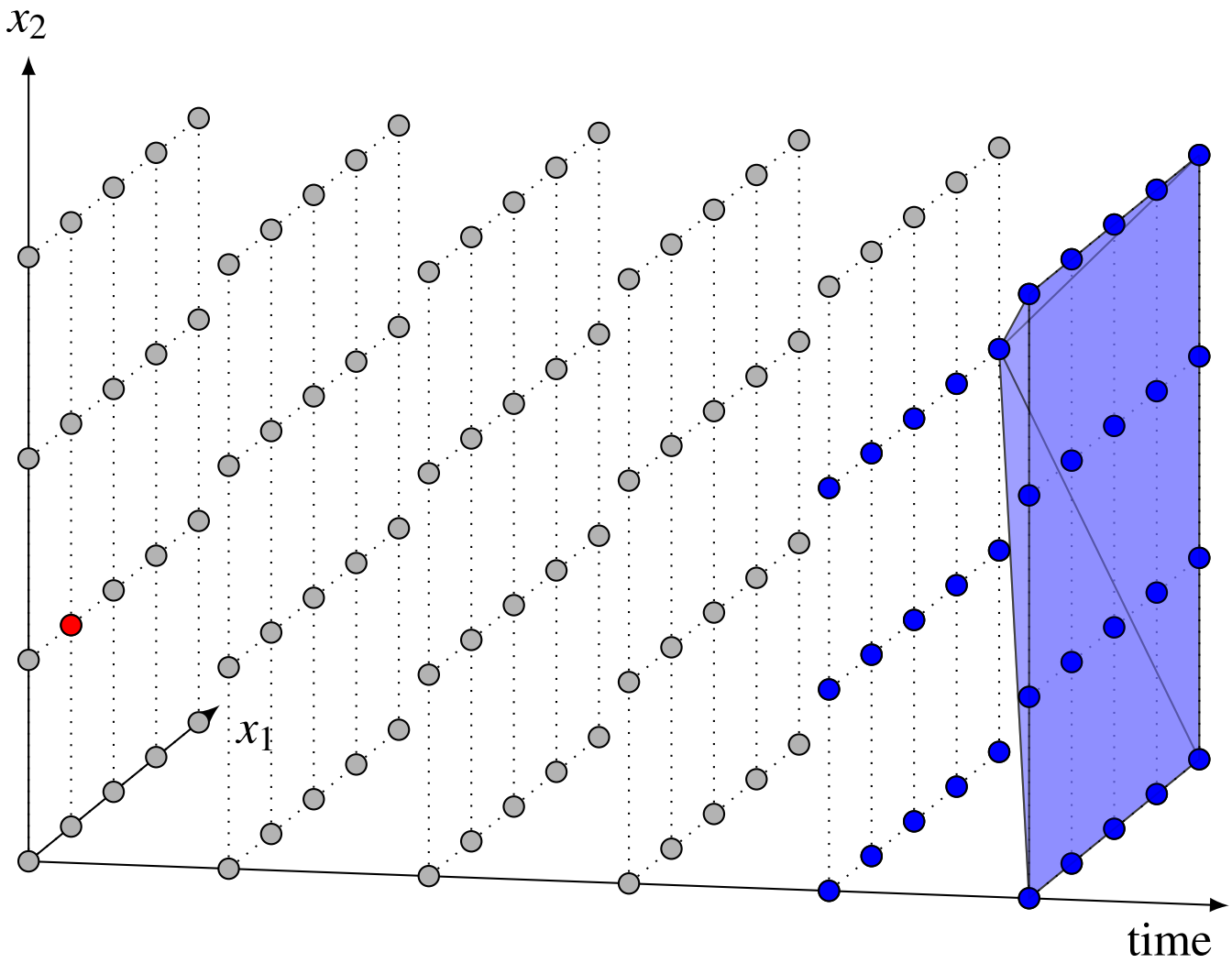
# Dynamic Programming: Illustration



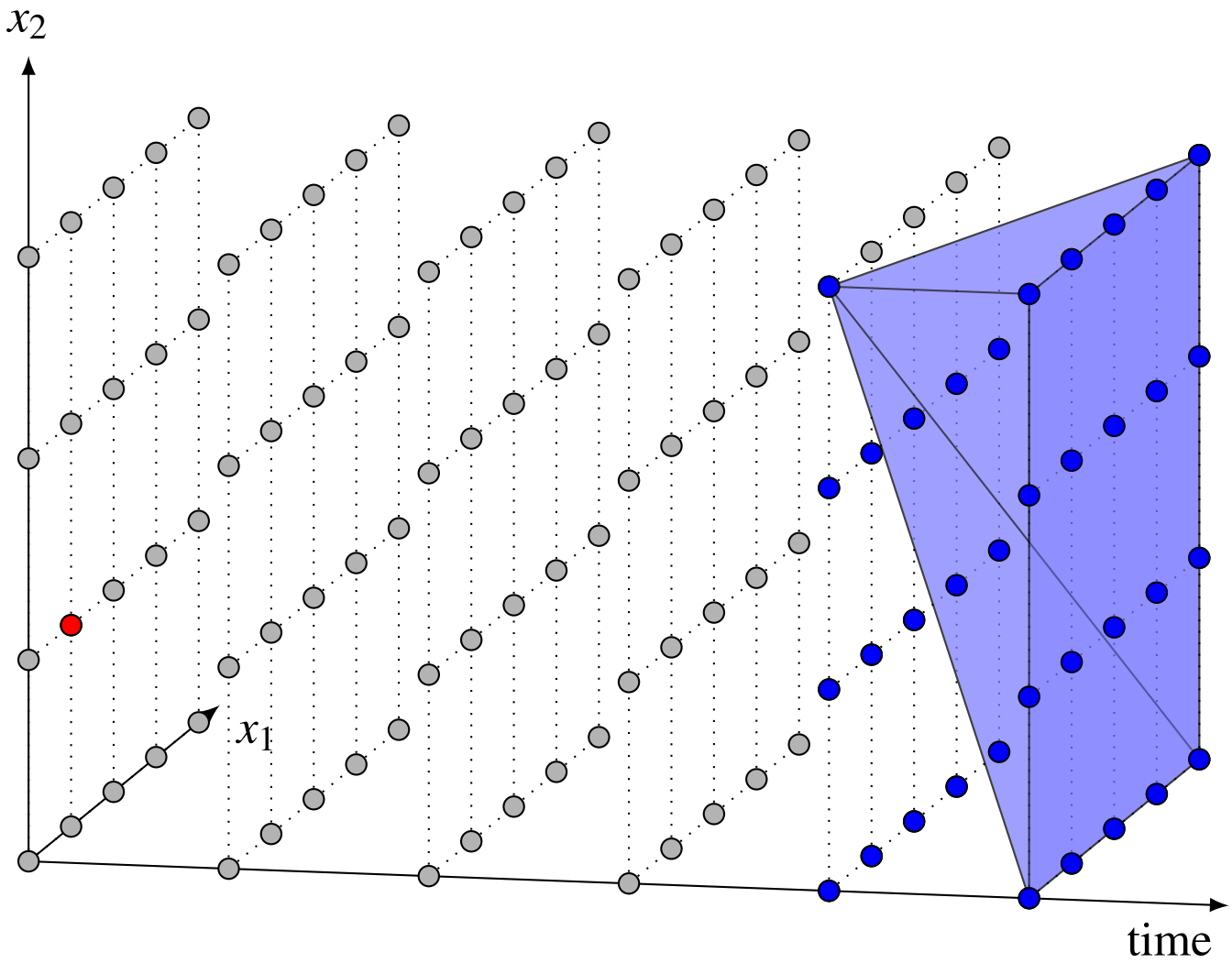
# Dynamic Programming: Illustration



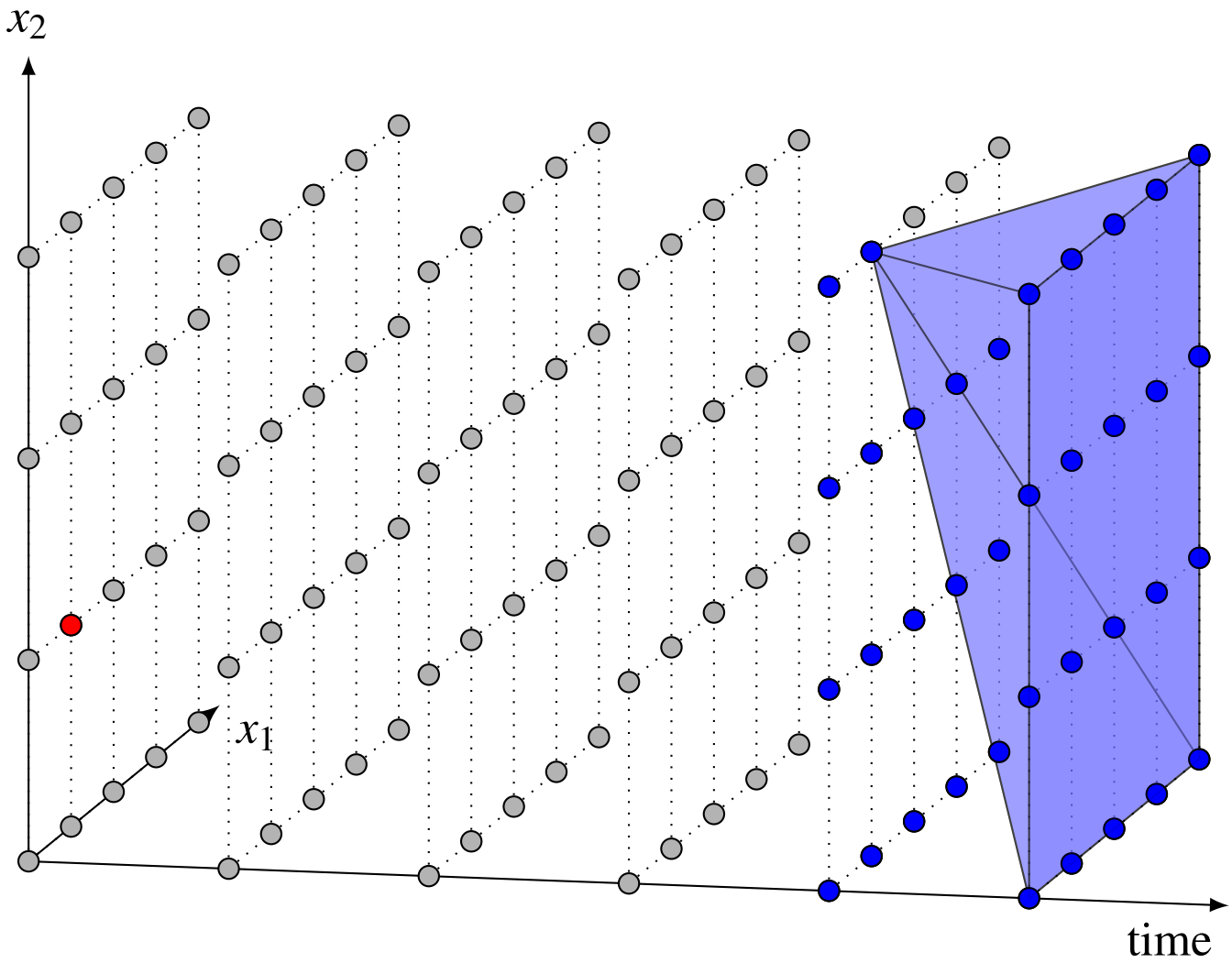
# Dynamic Programming: Illustration



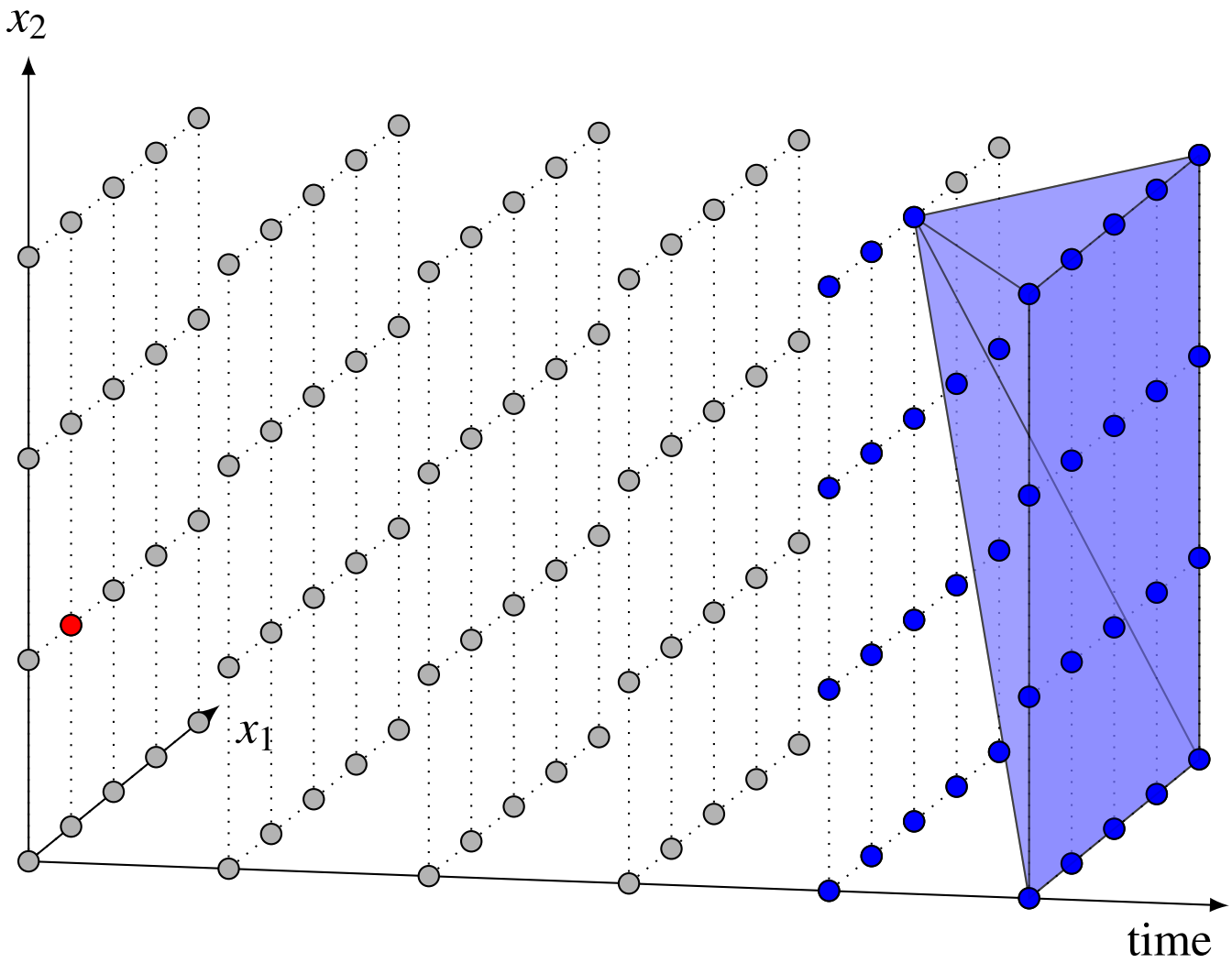
# Dynamic Programming: Illustration



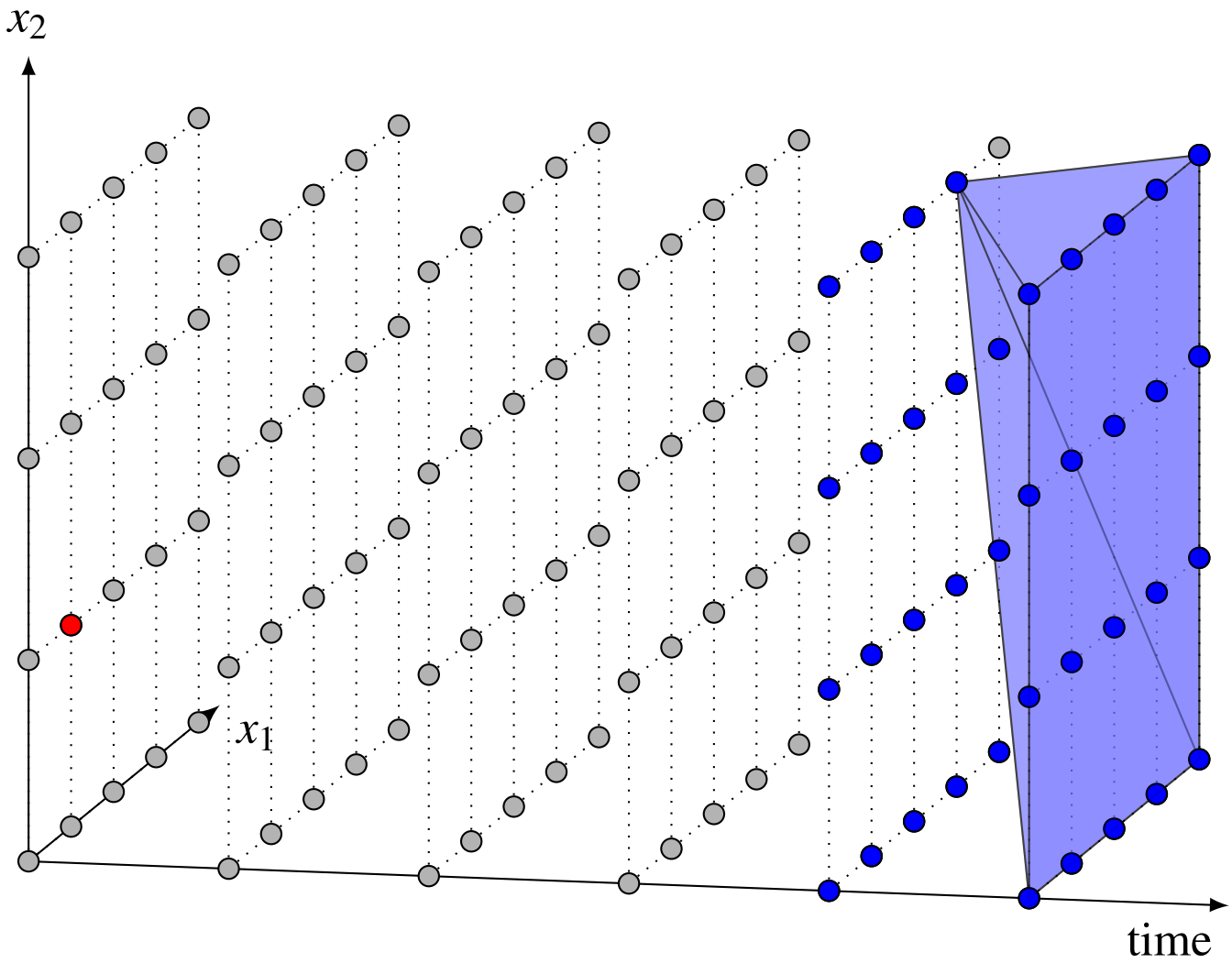
# Dynamic Programming: Illustration



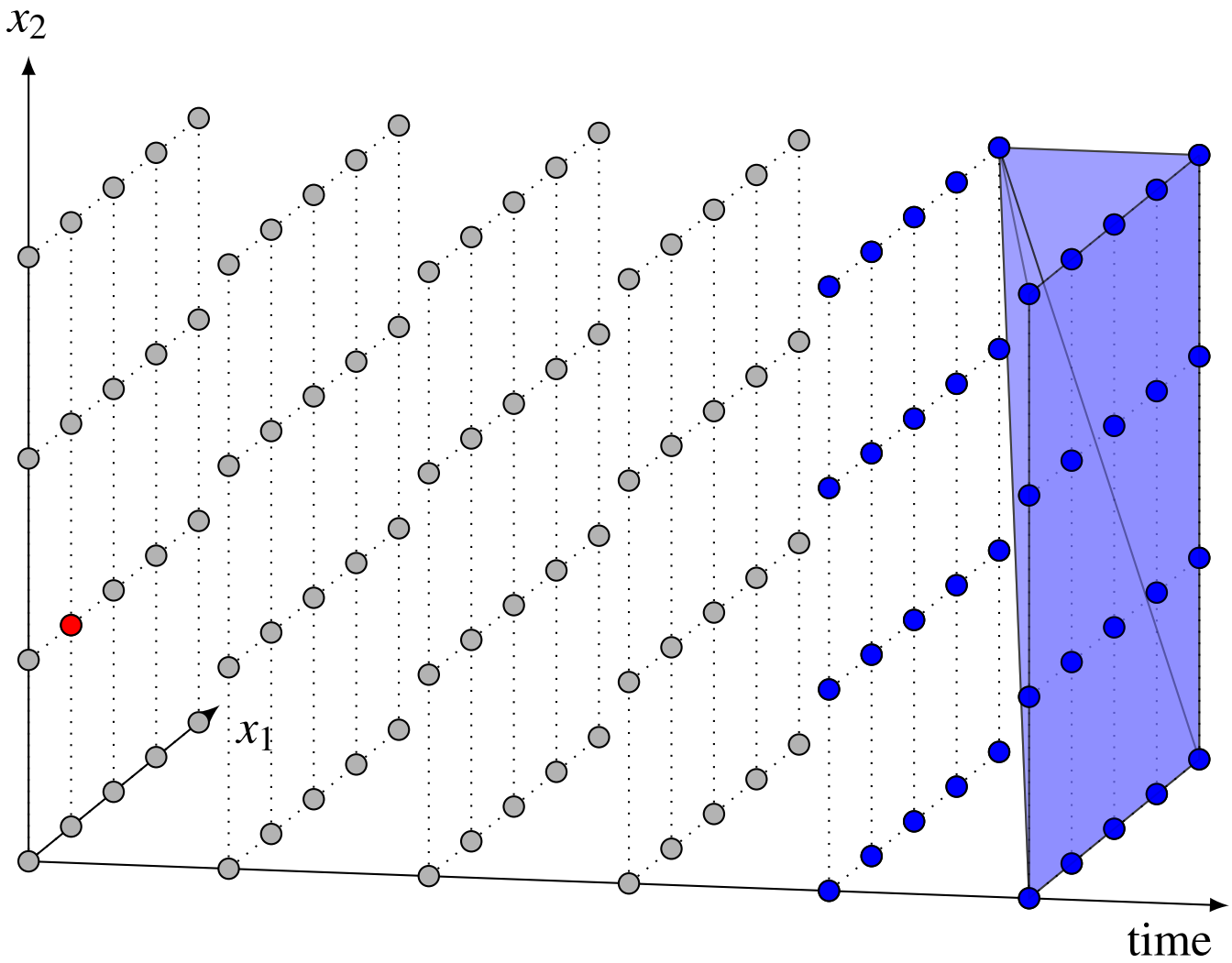
# Dynamic Programming: Illustration



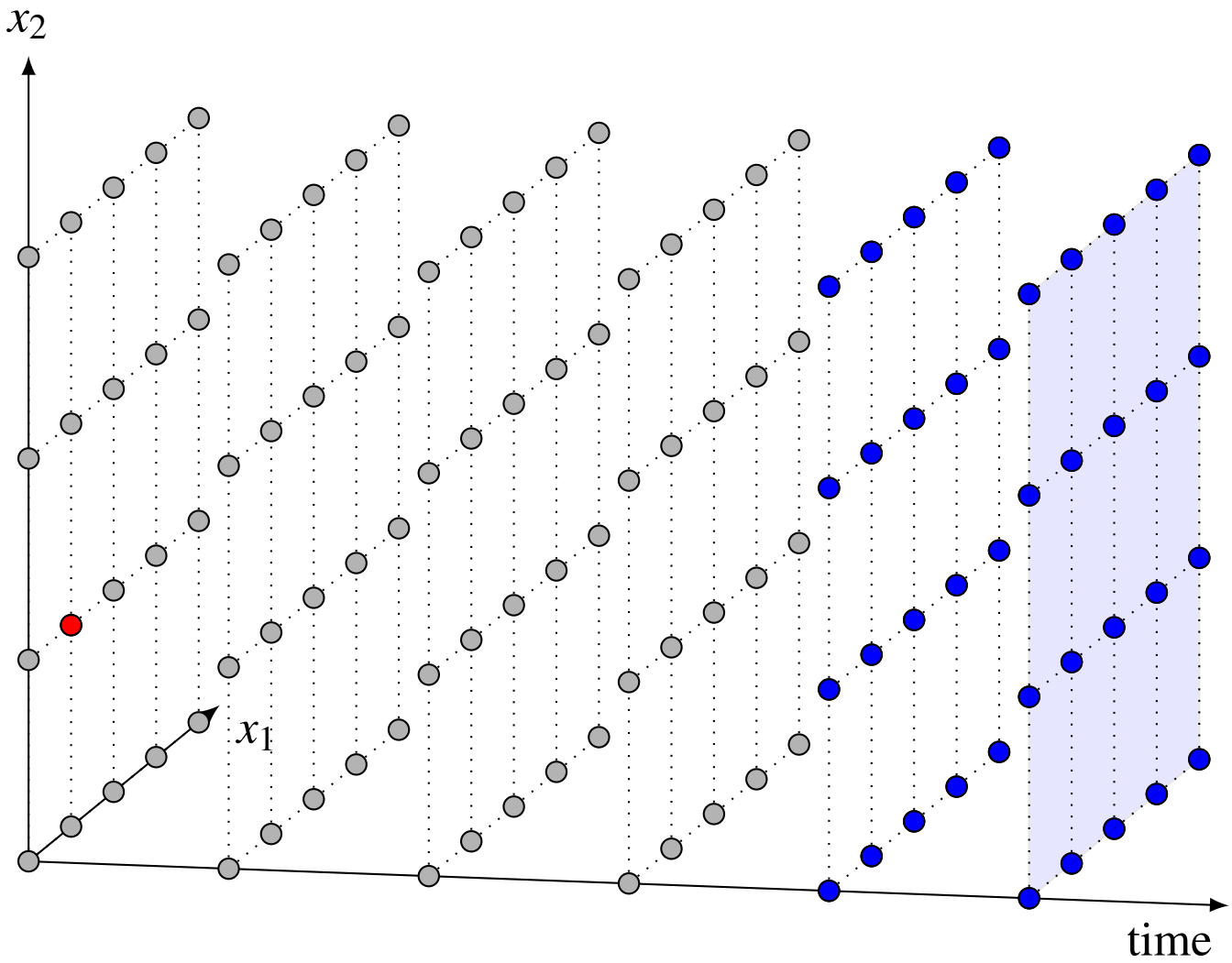
# Dynamic Programming: Illustration



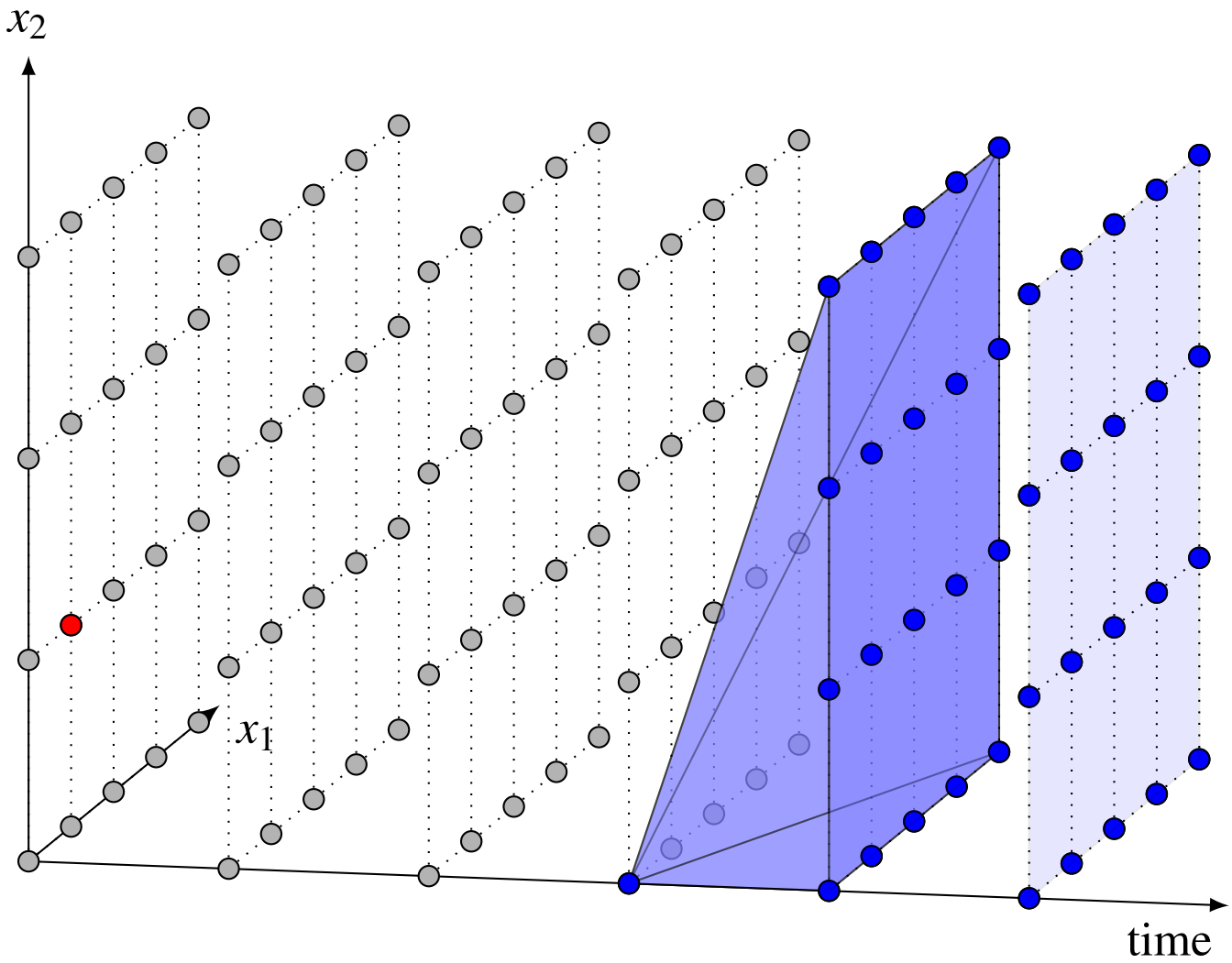
# Dynamic Programming: Illustration



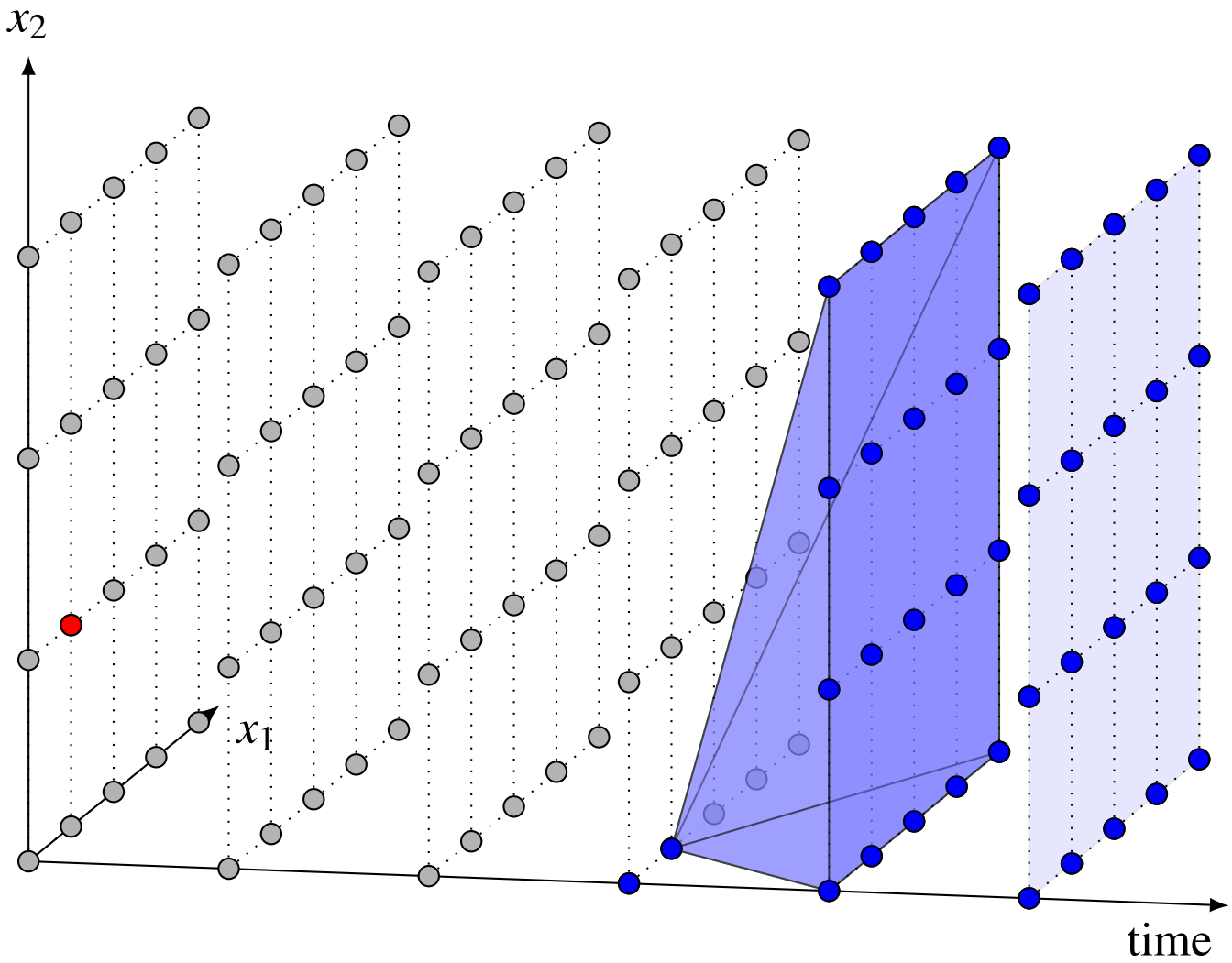
# Dynamic Programming: Illustration



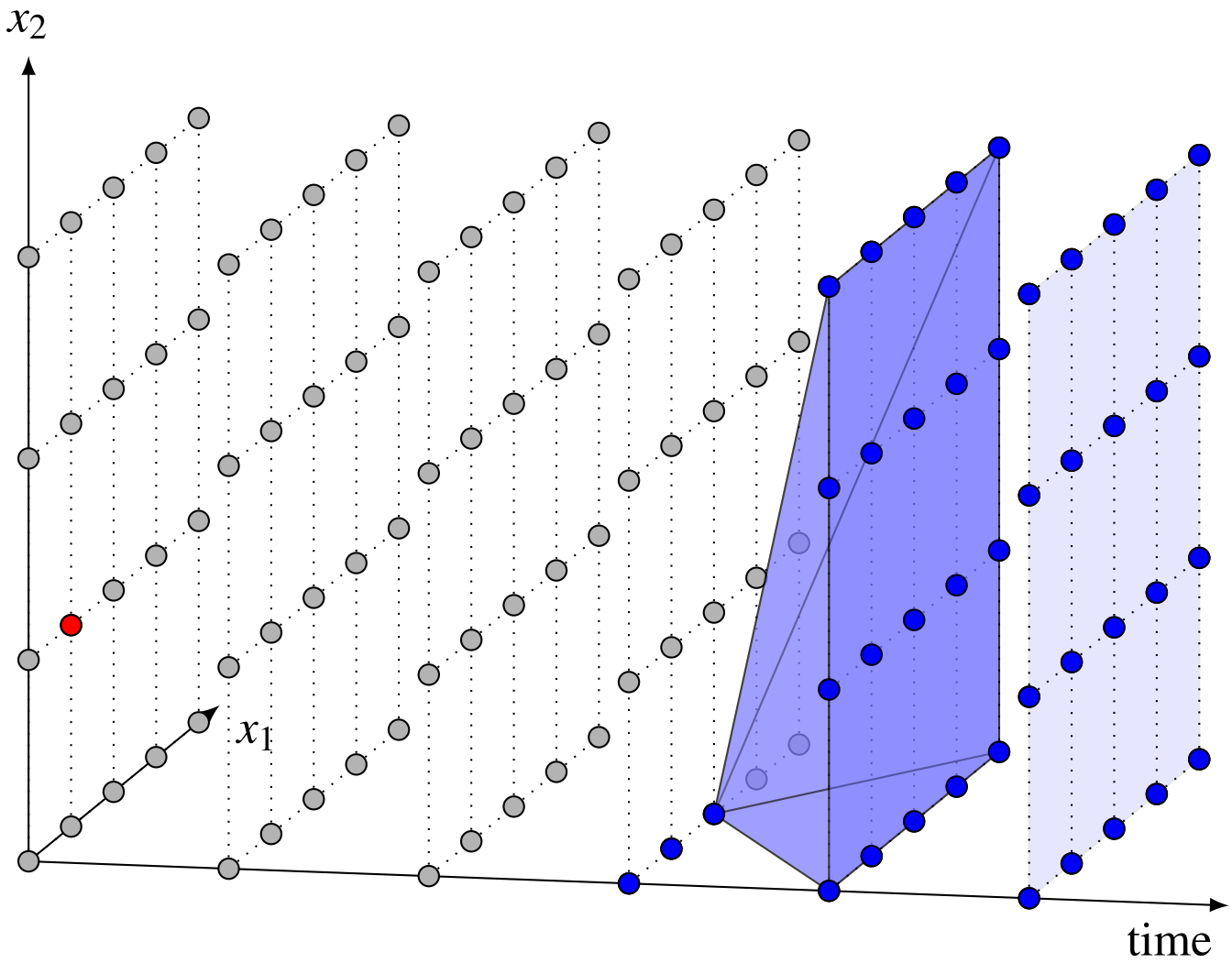
# Dynamic Programming: Illustration



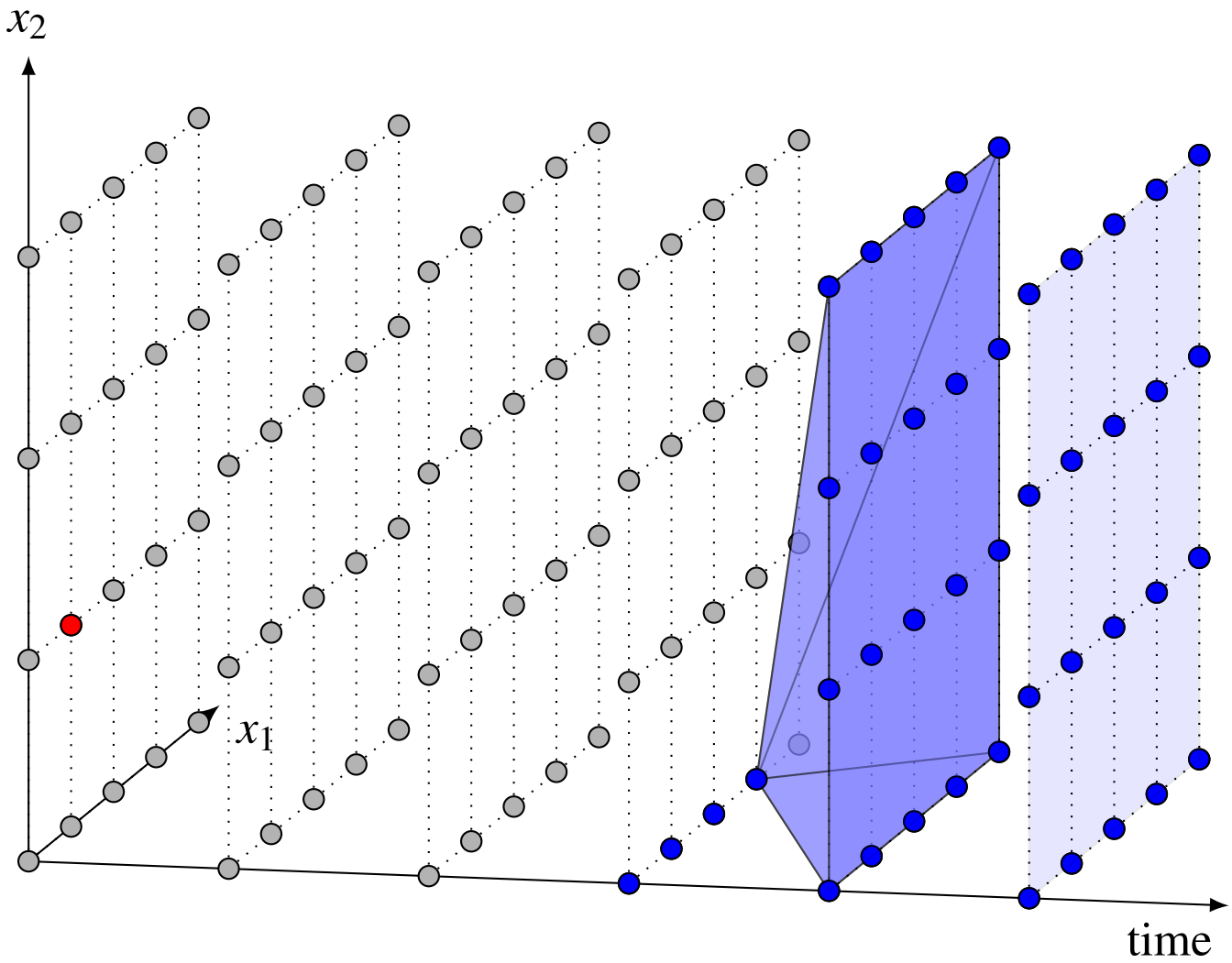
# Dynamic Programming: Illustration



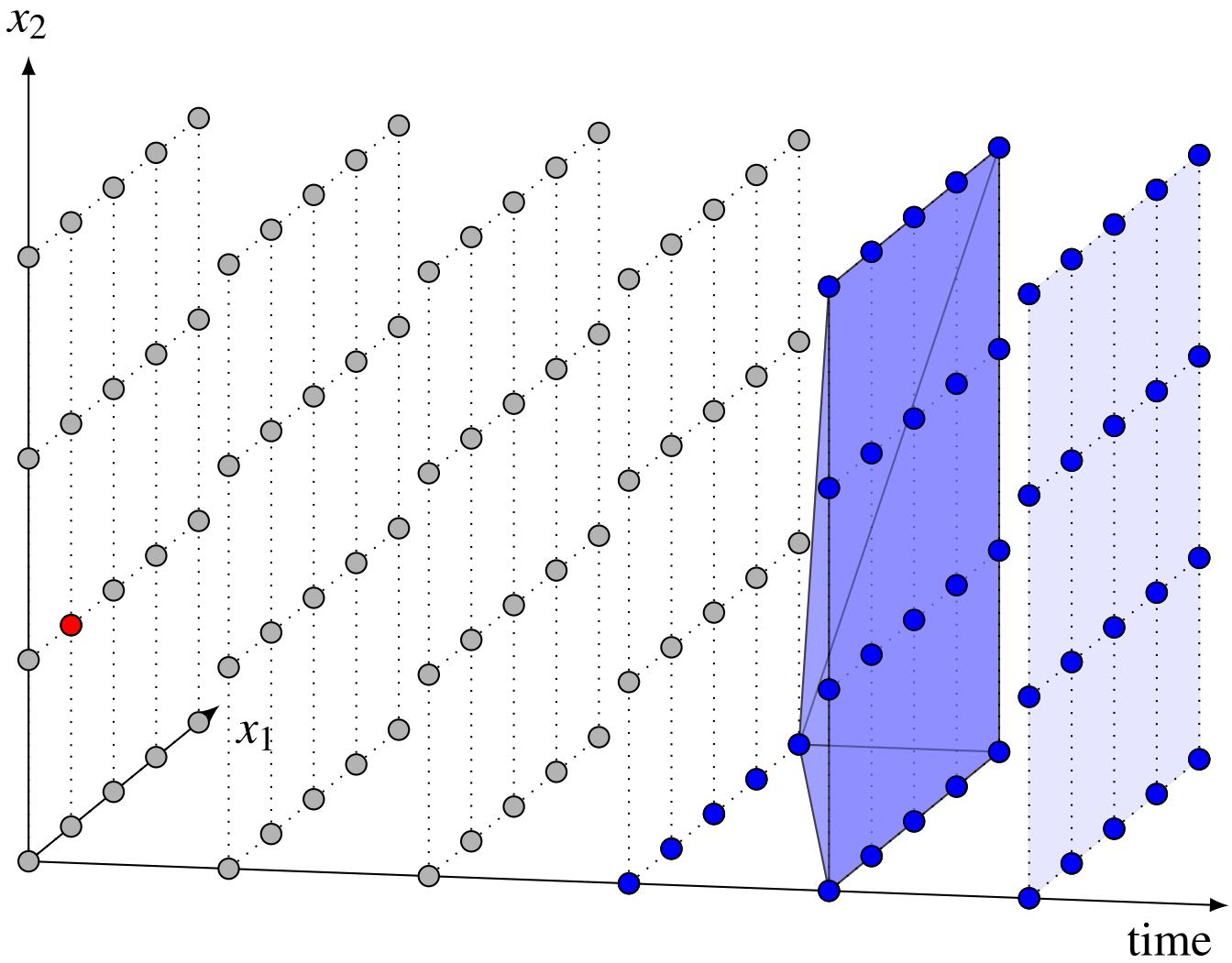
# Dynamic Programming: Illustration



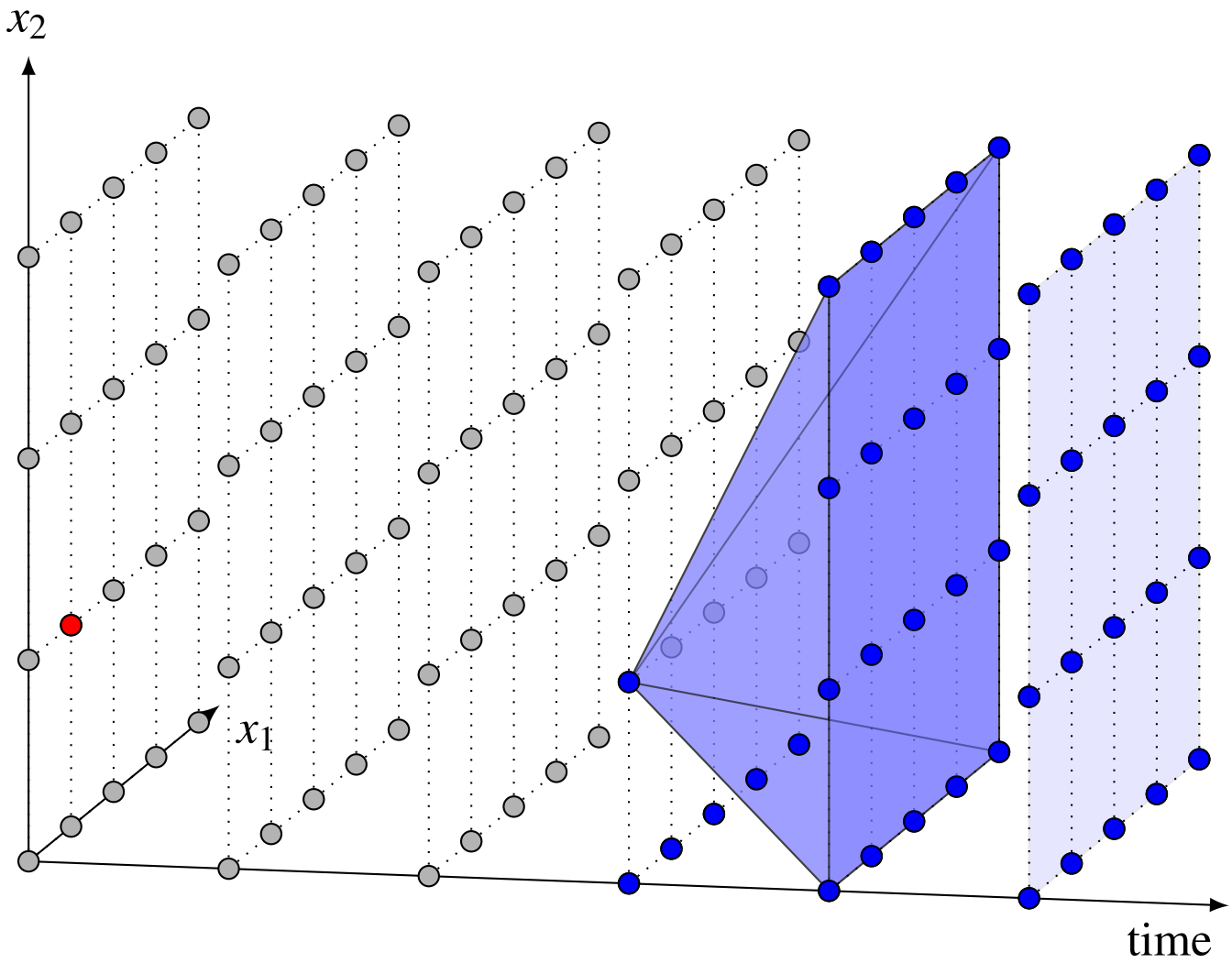
# Dynamic Programming: Illustration



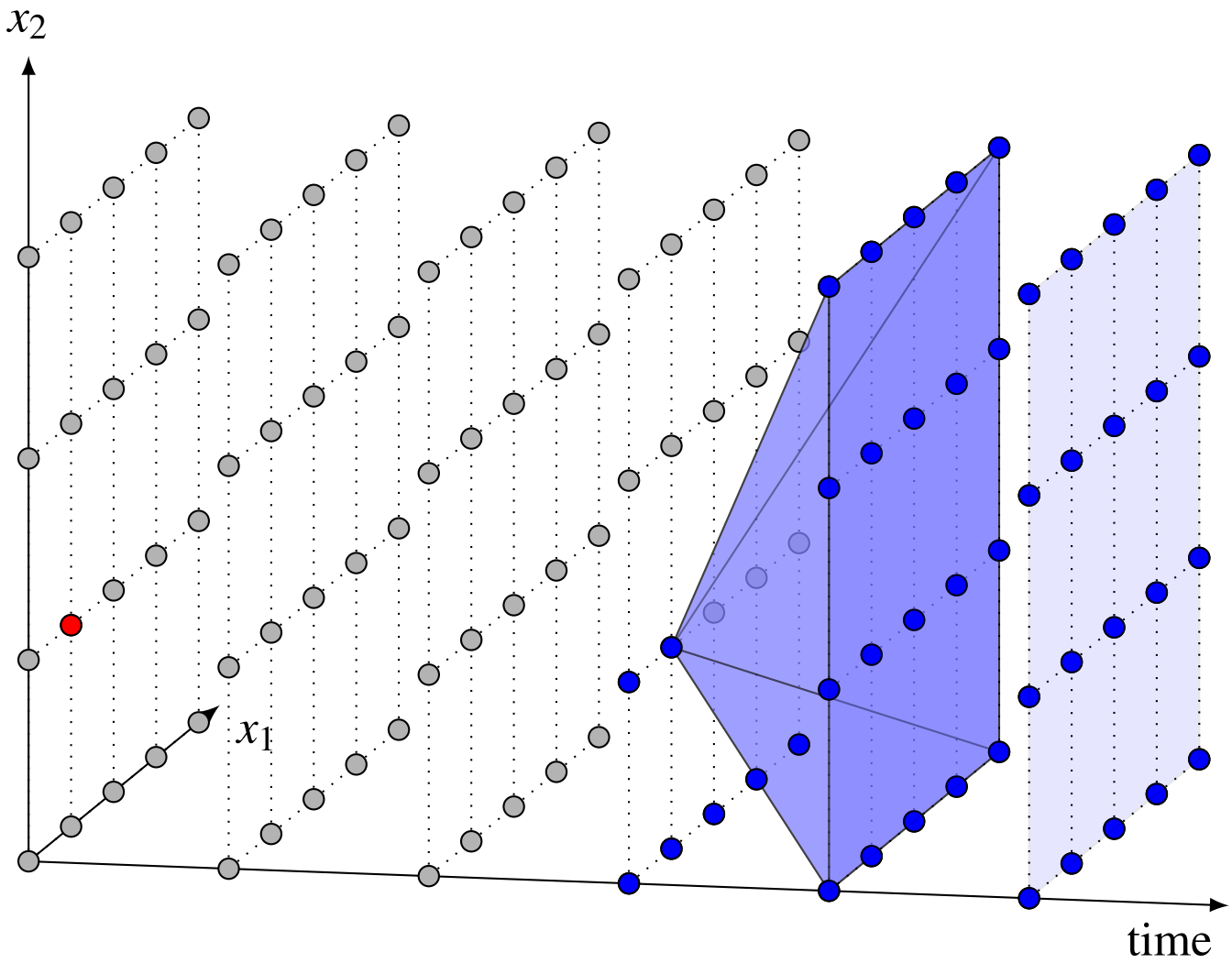
# Dynamic Programming: Illustration



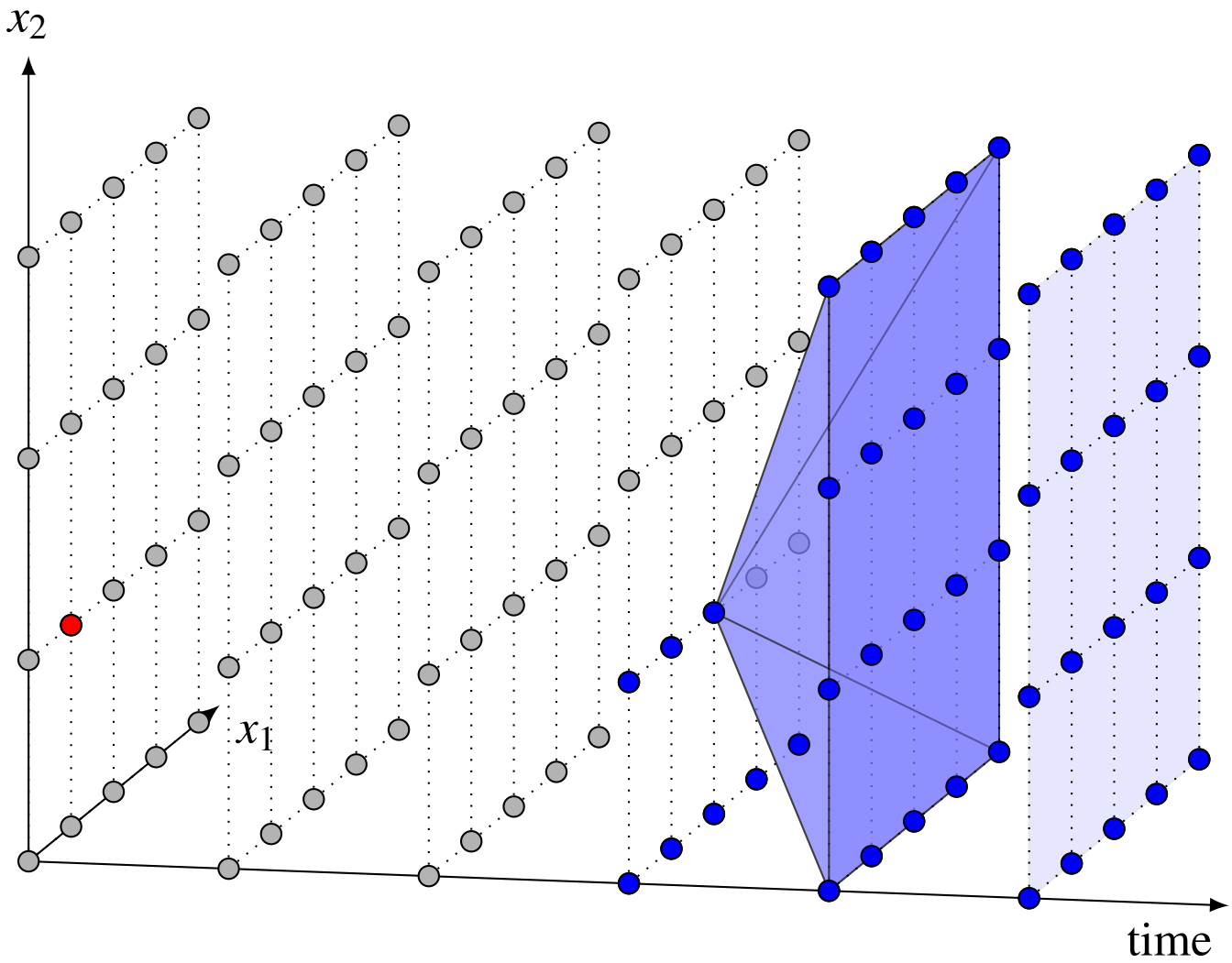
# Dynamic Programming: Illustration



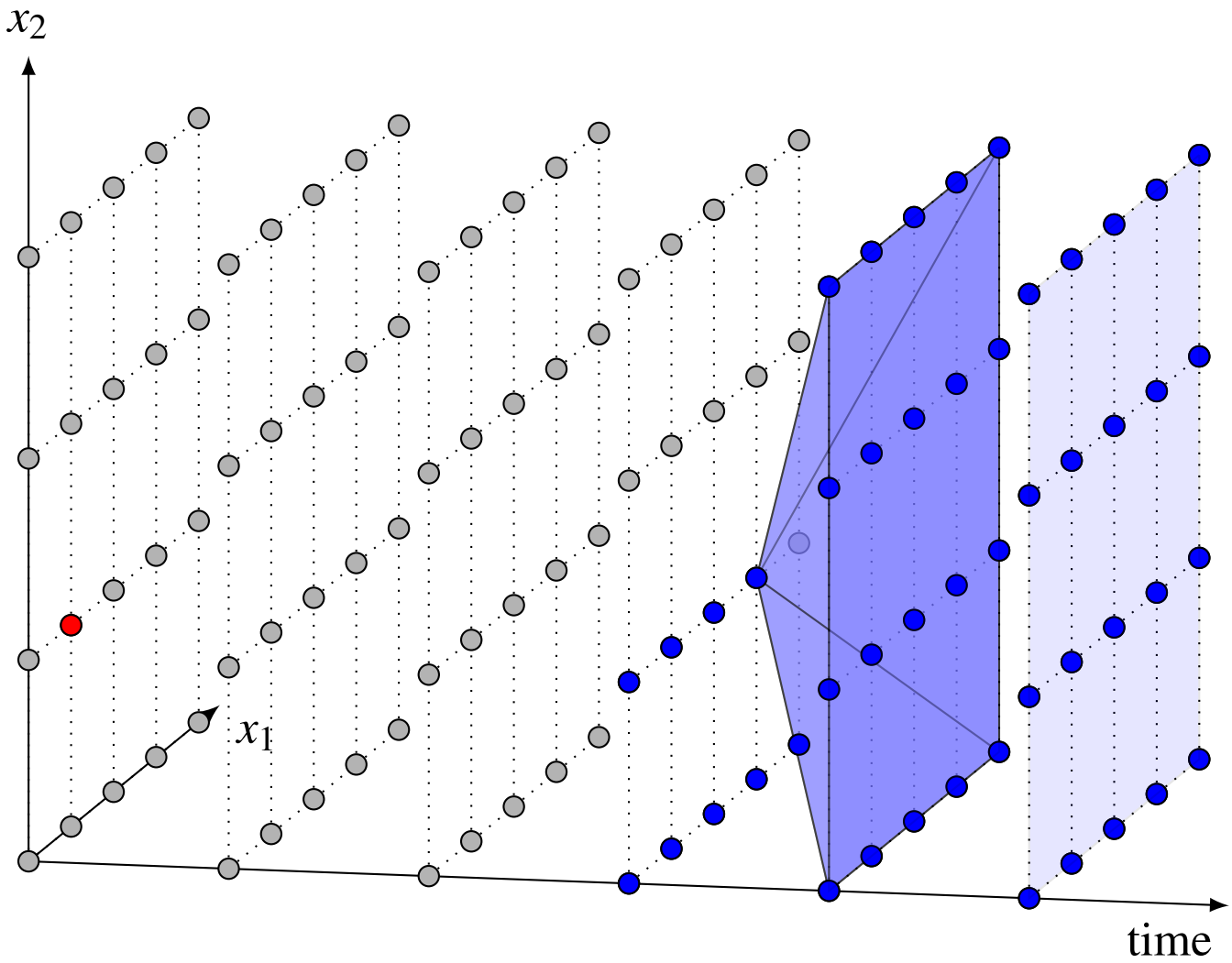
# Dynamic Programming: Illustration



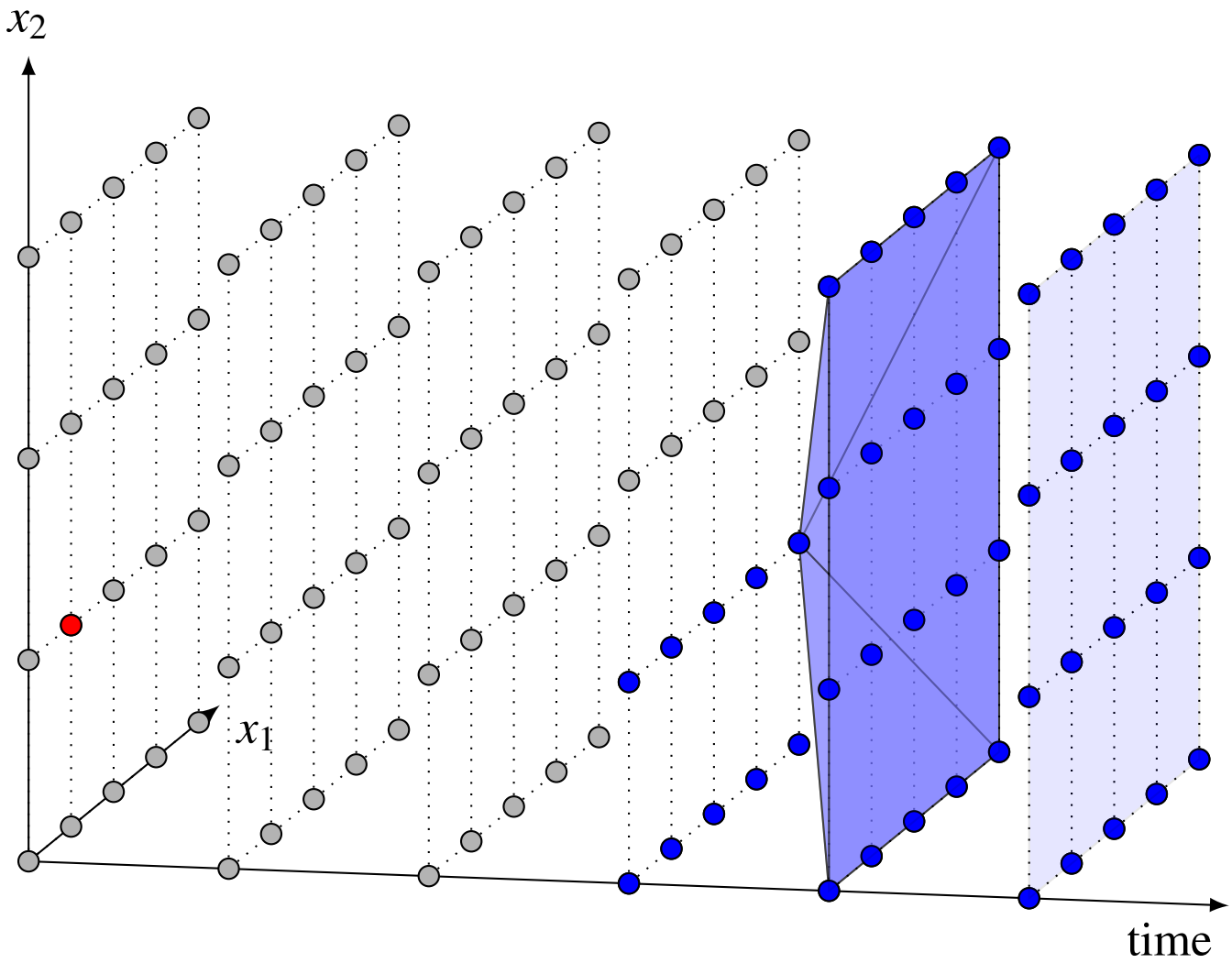
# Dynamic Programming: Illustration



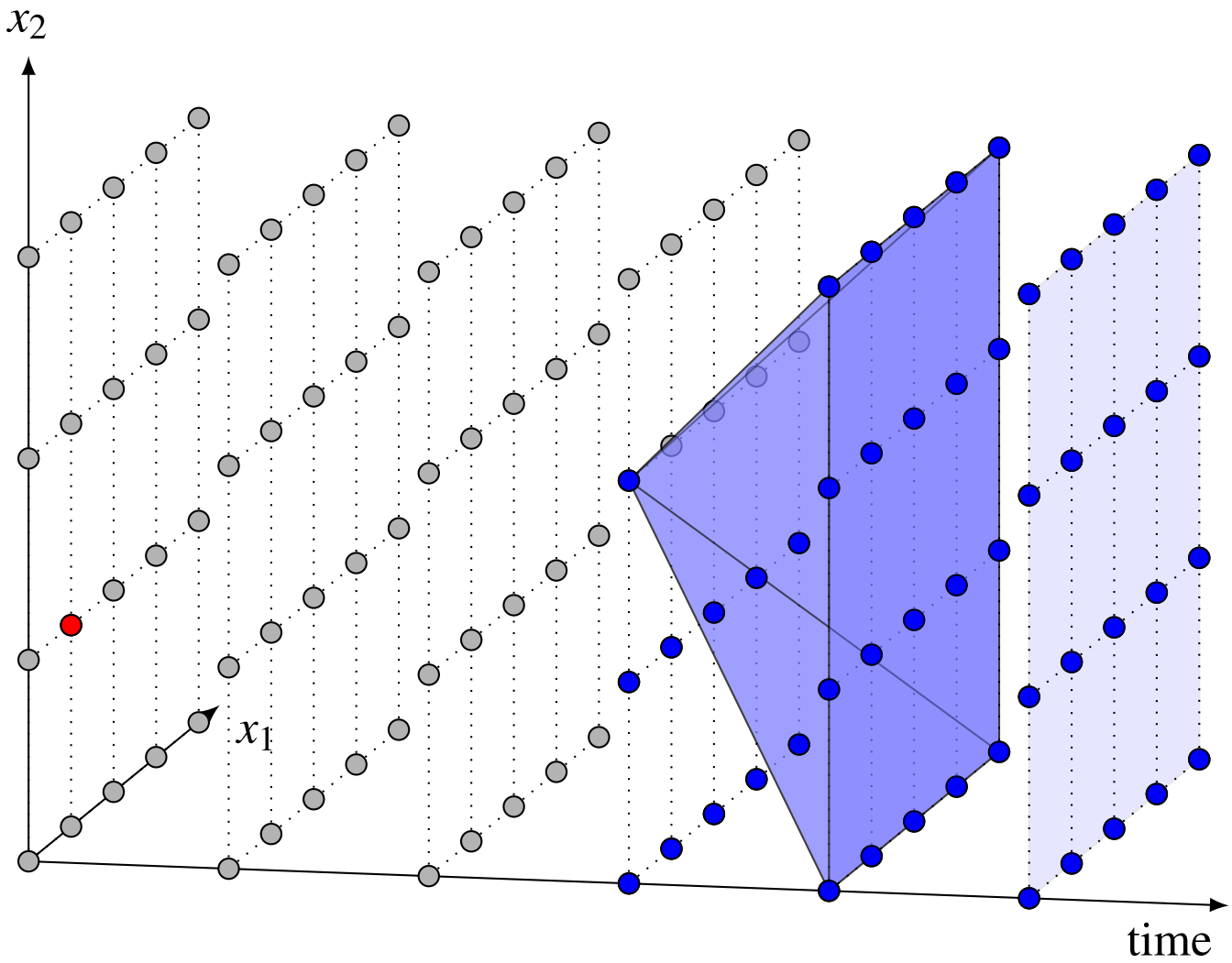
# Dynamic Programming: Illustration



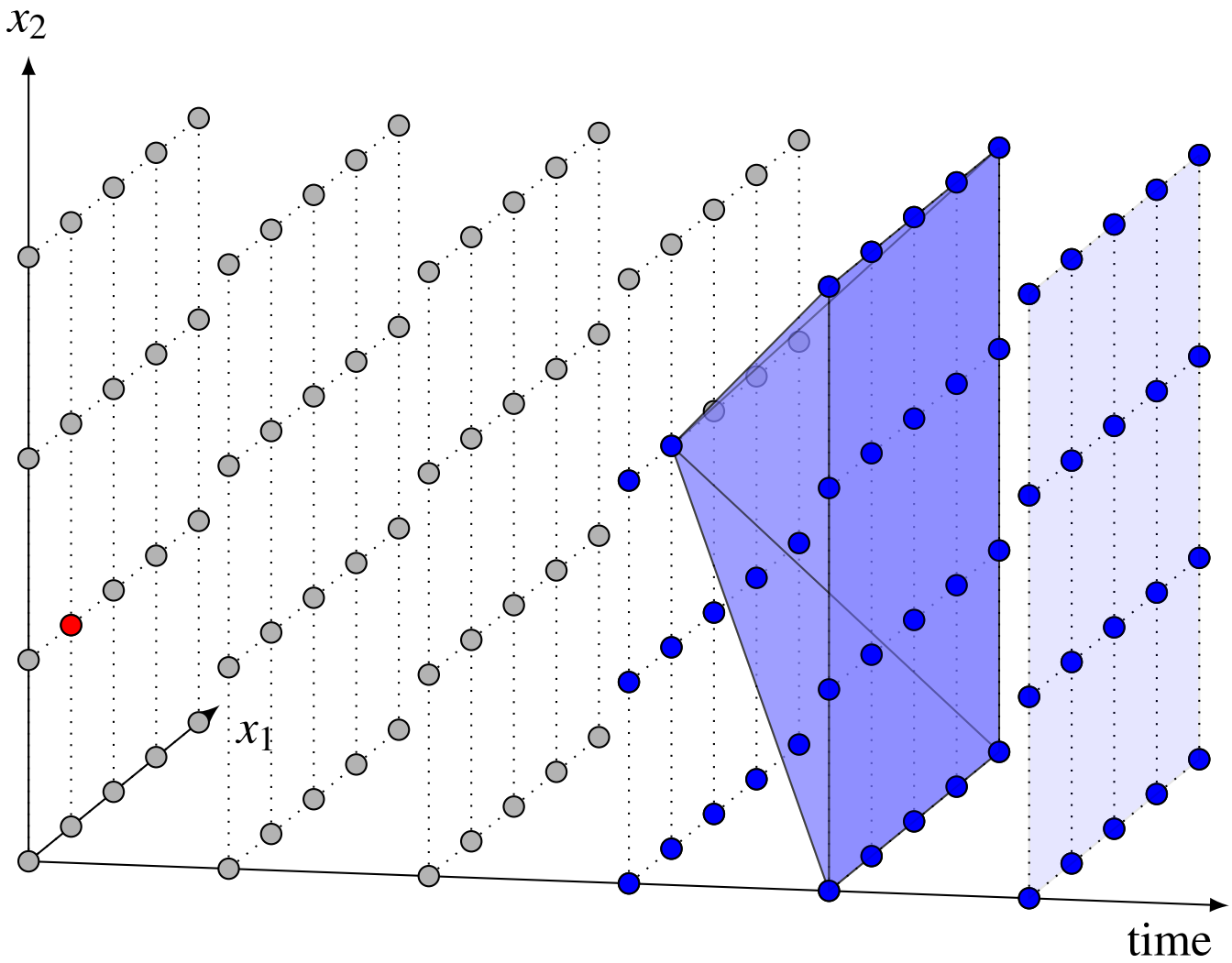
# Dynamic Programming: Illustration



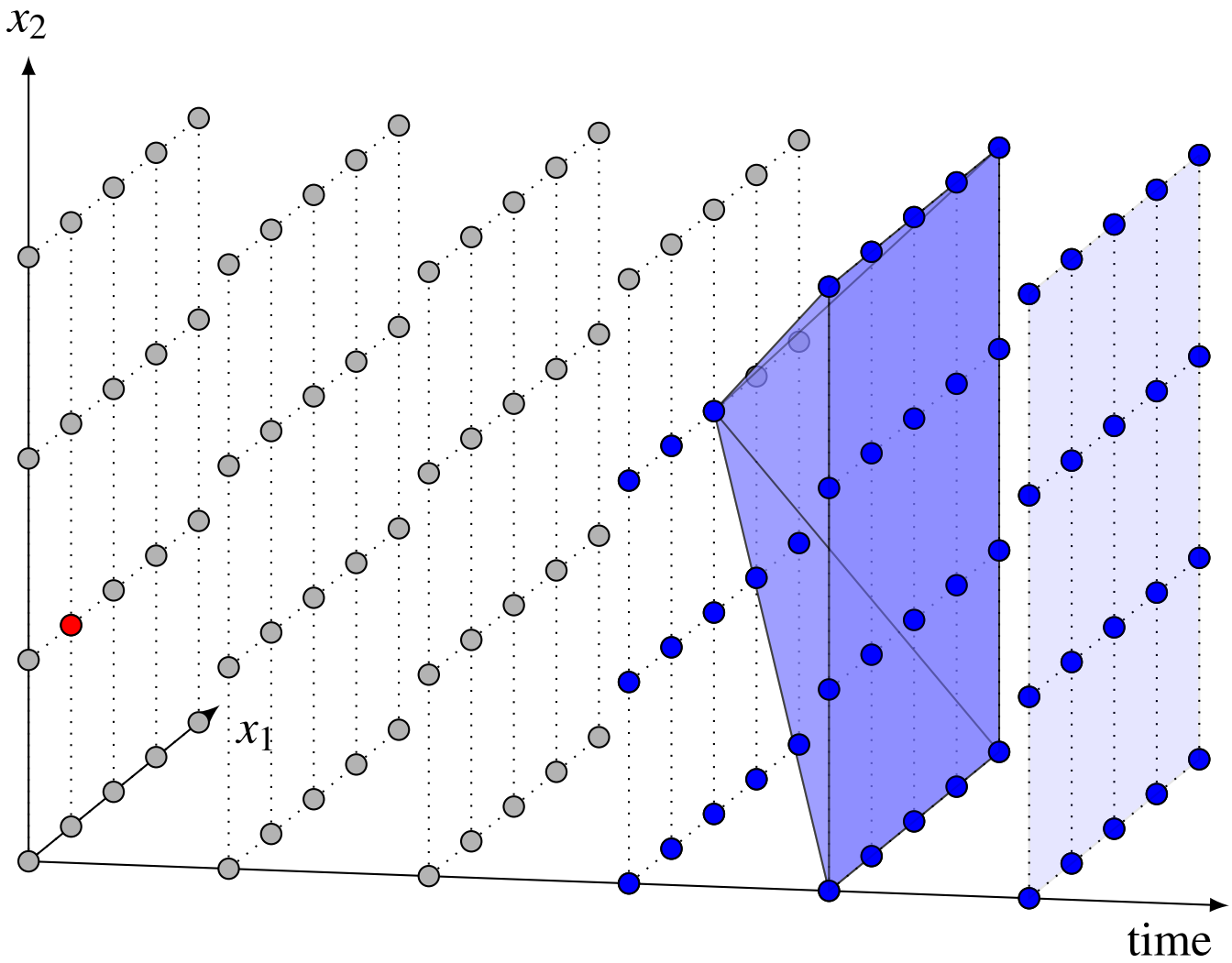
# Dynamic Programming: Illustration



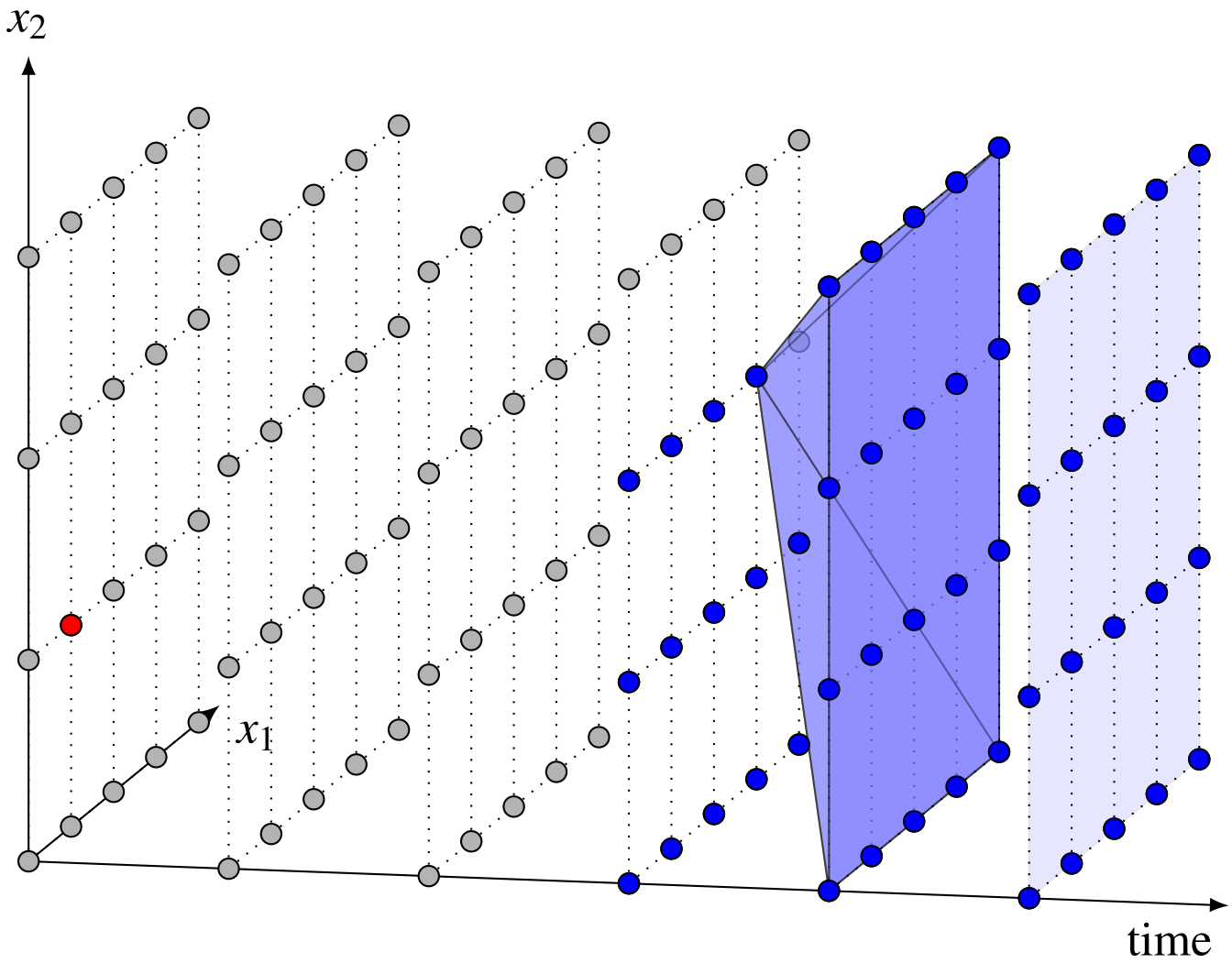
# Dynamic Programming: Illustration



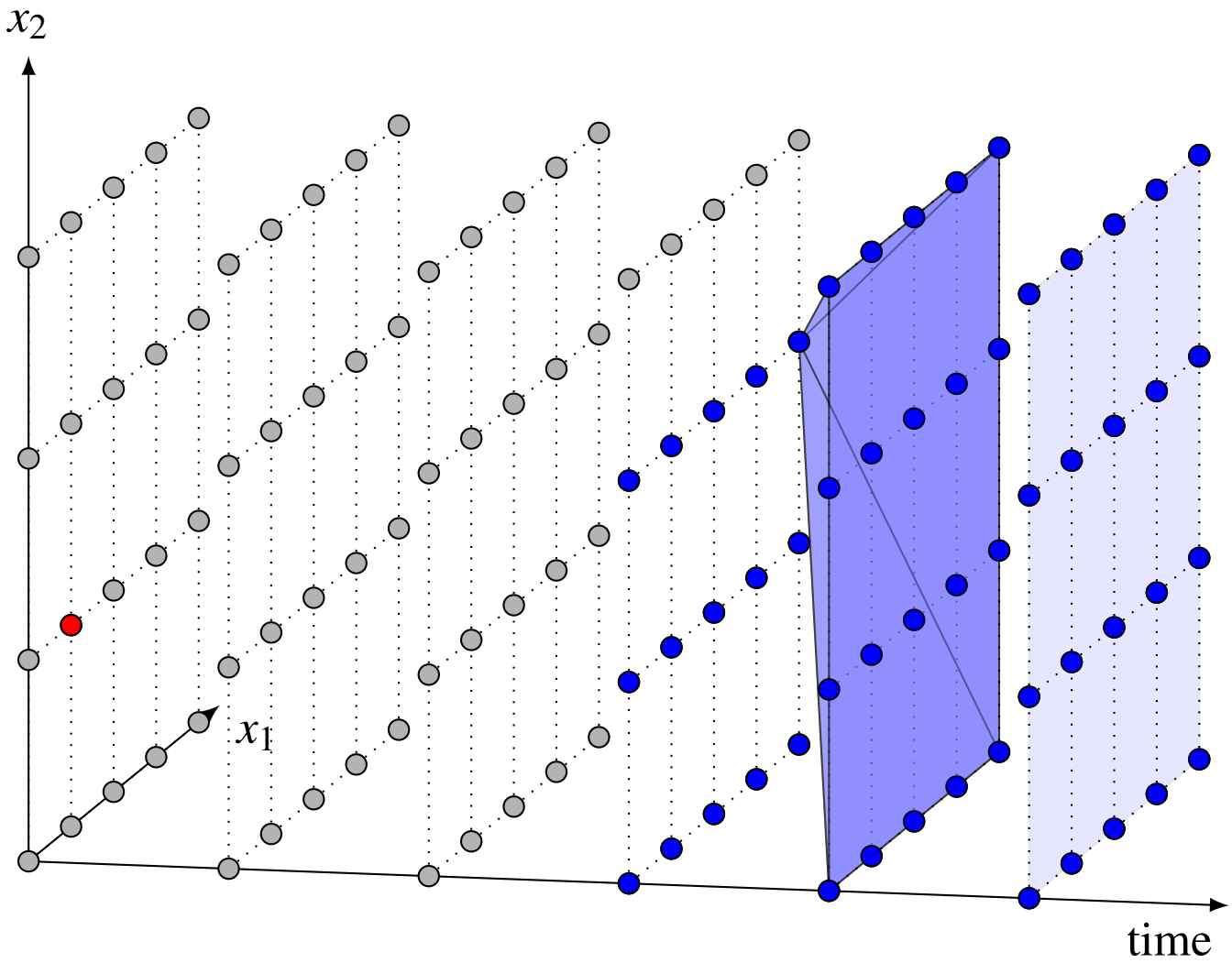
# Dynamic Programming: Illustration



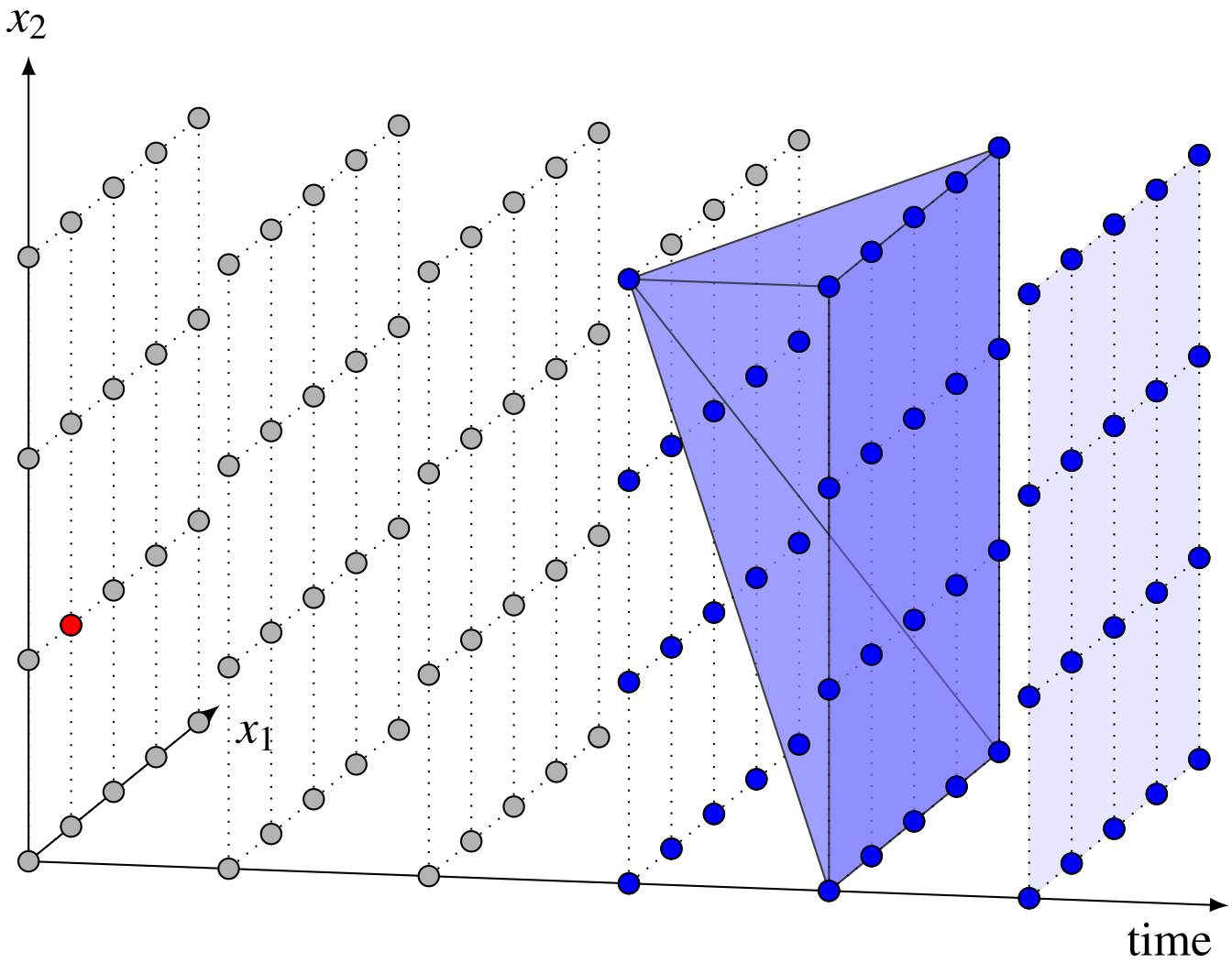
# Dynamic Programming: Illustration



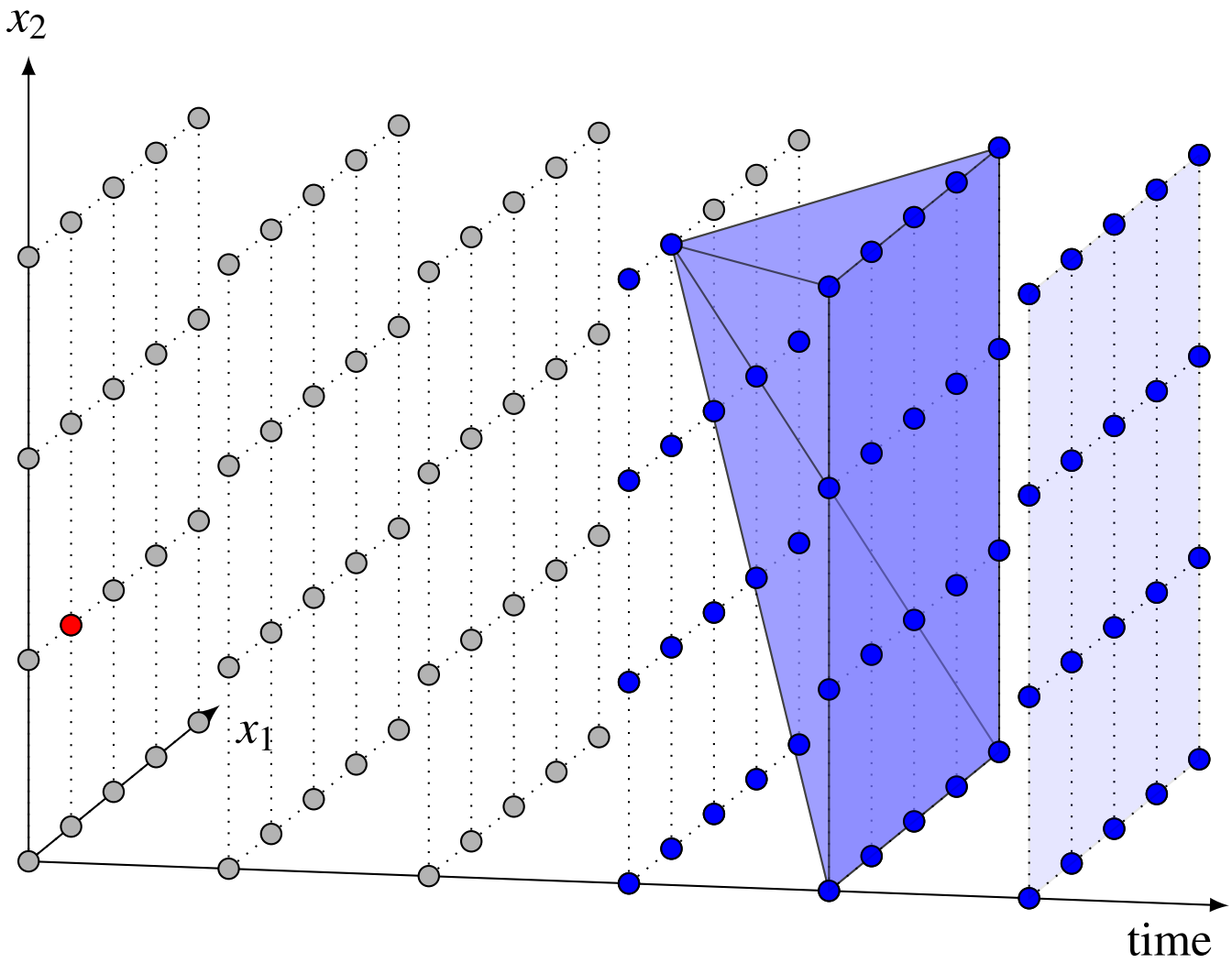
# Dynamic Programming: Illustration



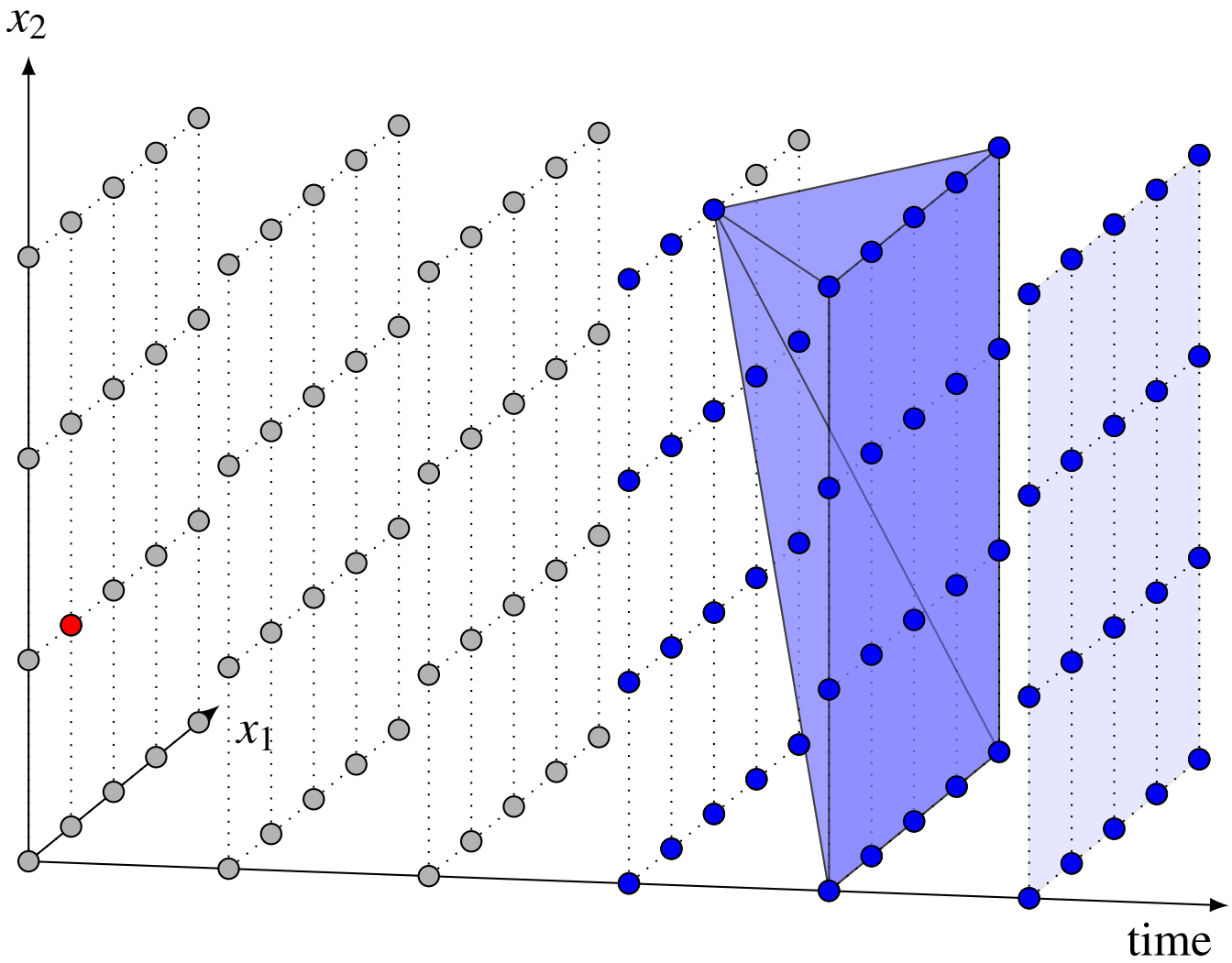
# Dynamic Programming: Illustration



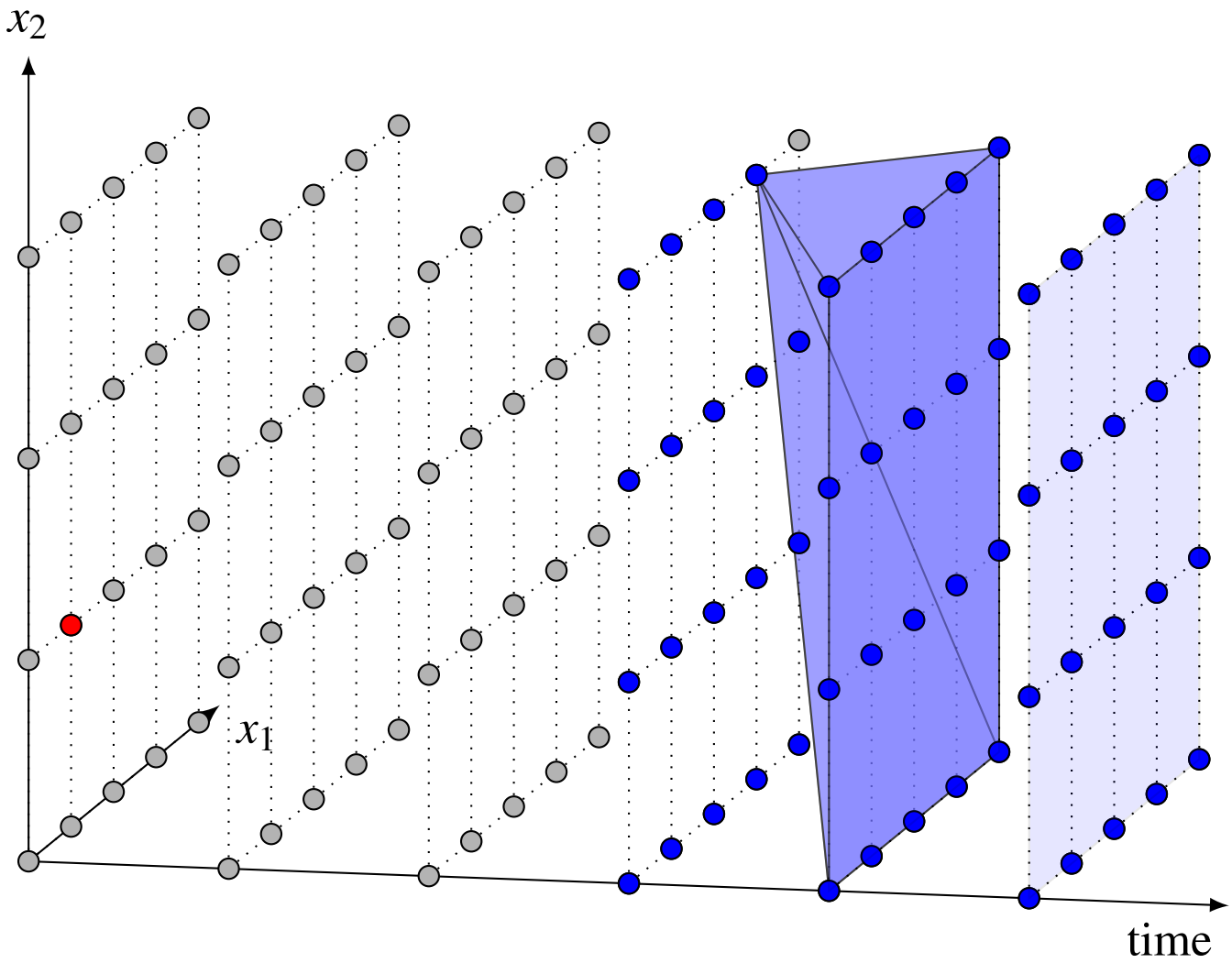
# Dynamic Programming: Illustration



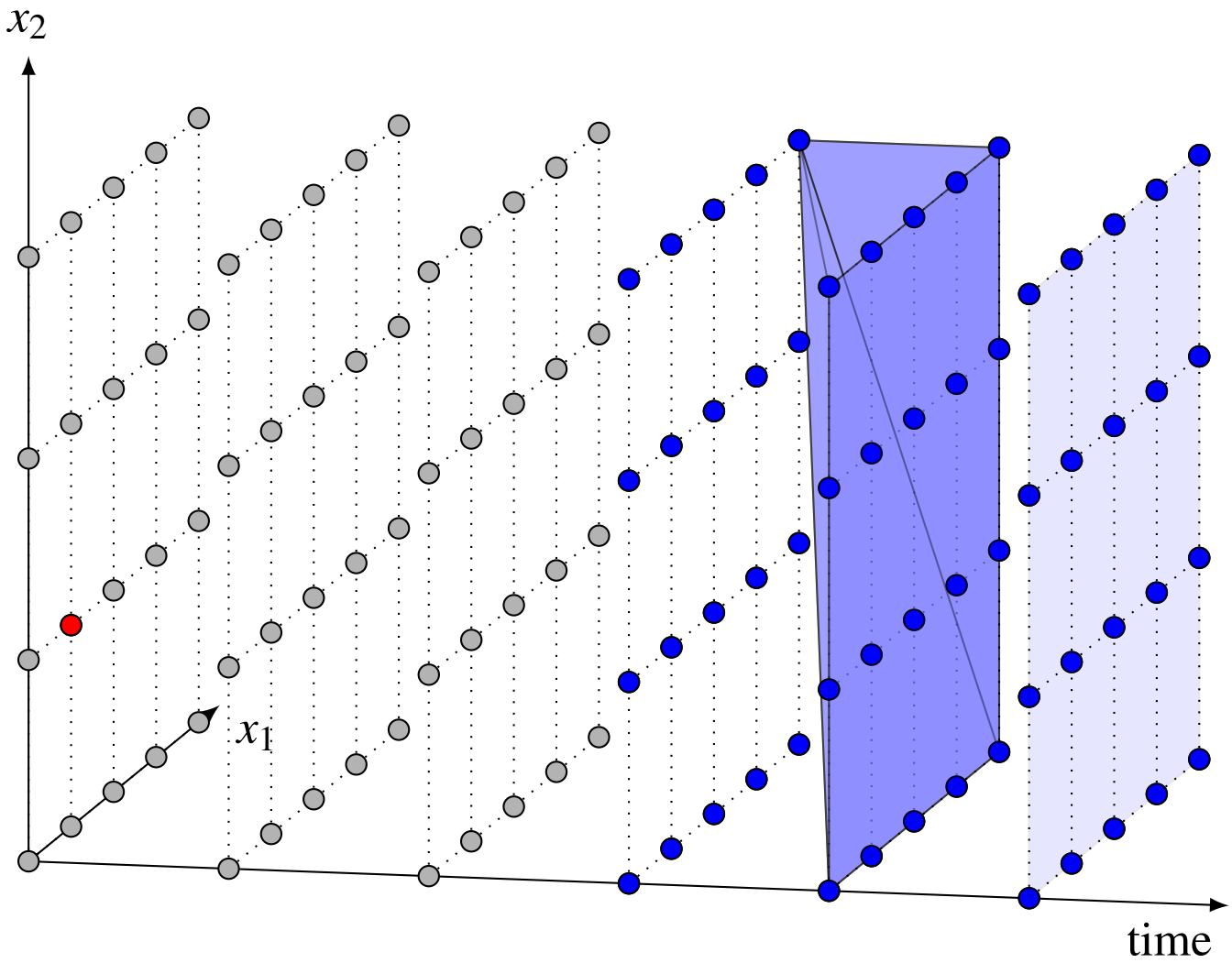
# Dynamic Programming: Illustration



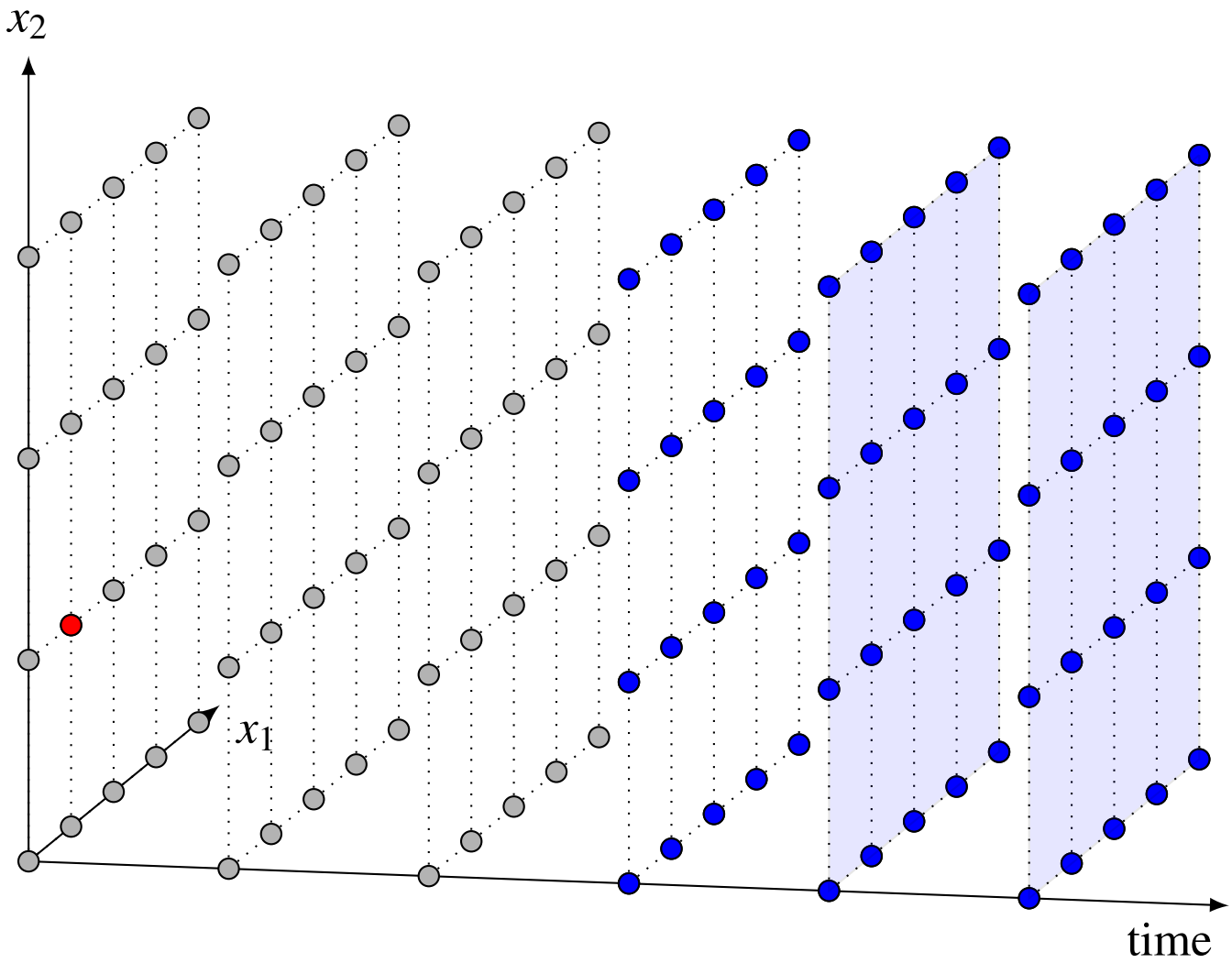
# Dynamic Programming: Illustration



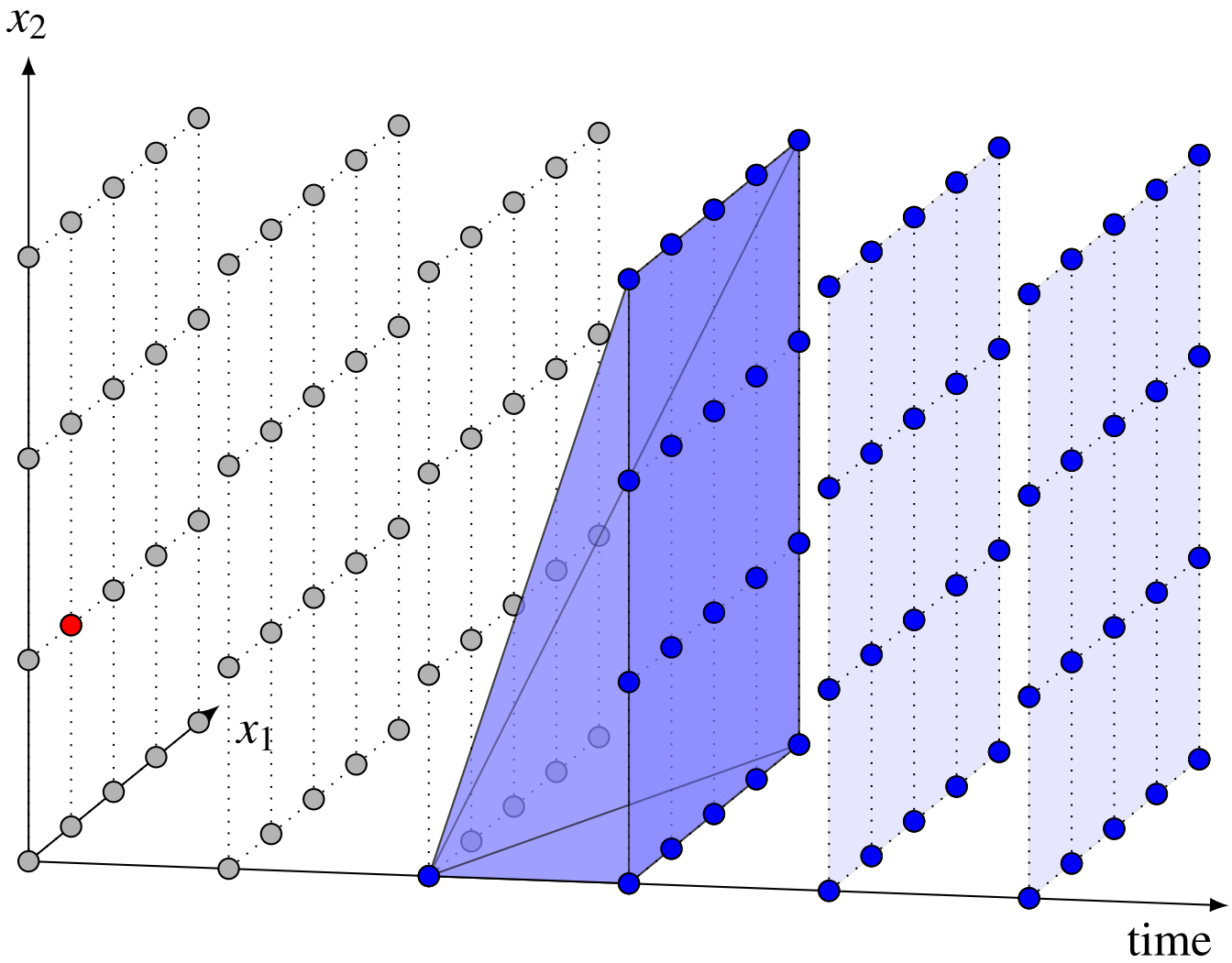
# Dynamic Programming: Illustration



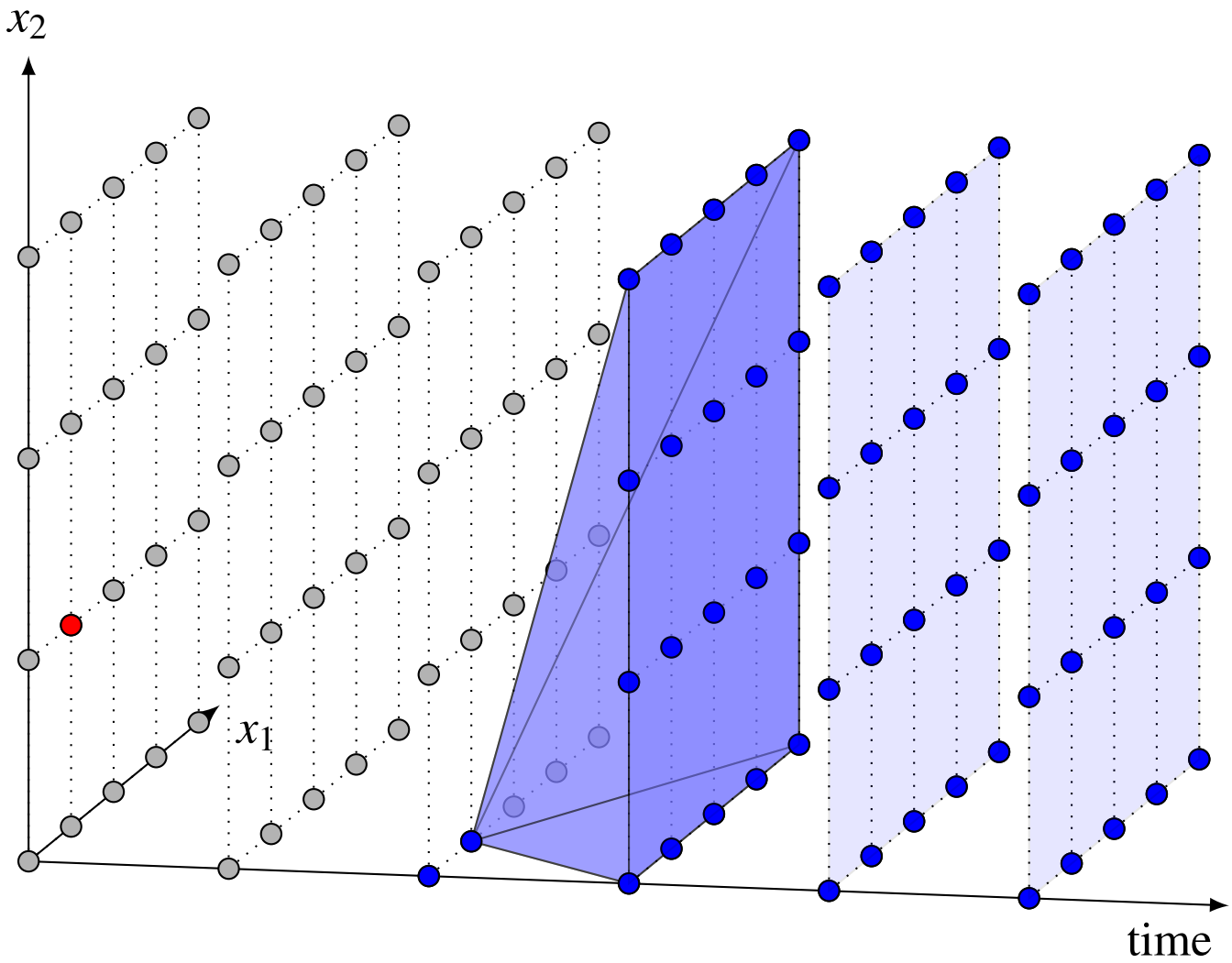
# Dynamic Programming: Illustration



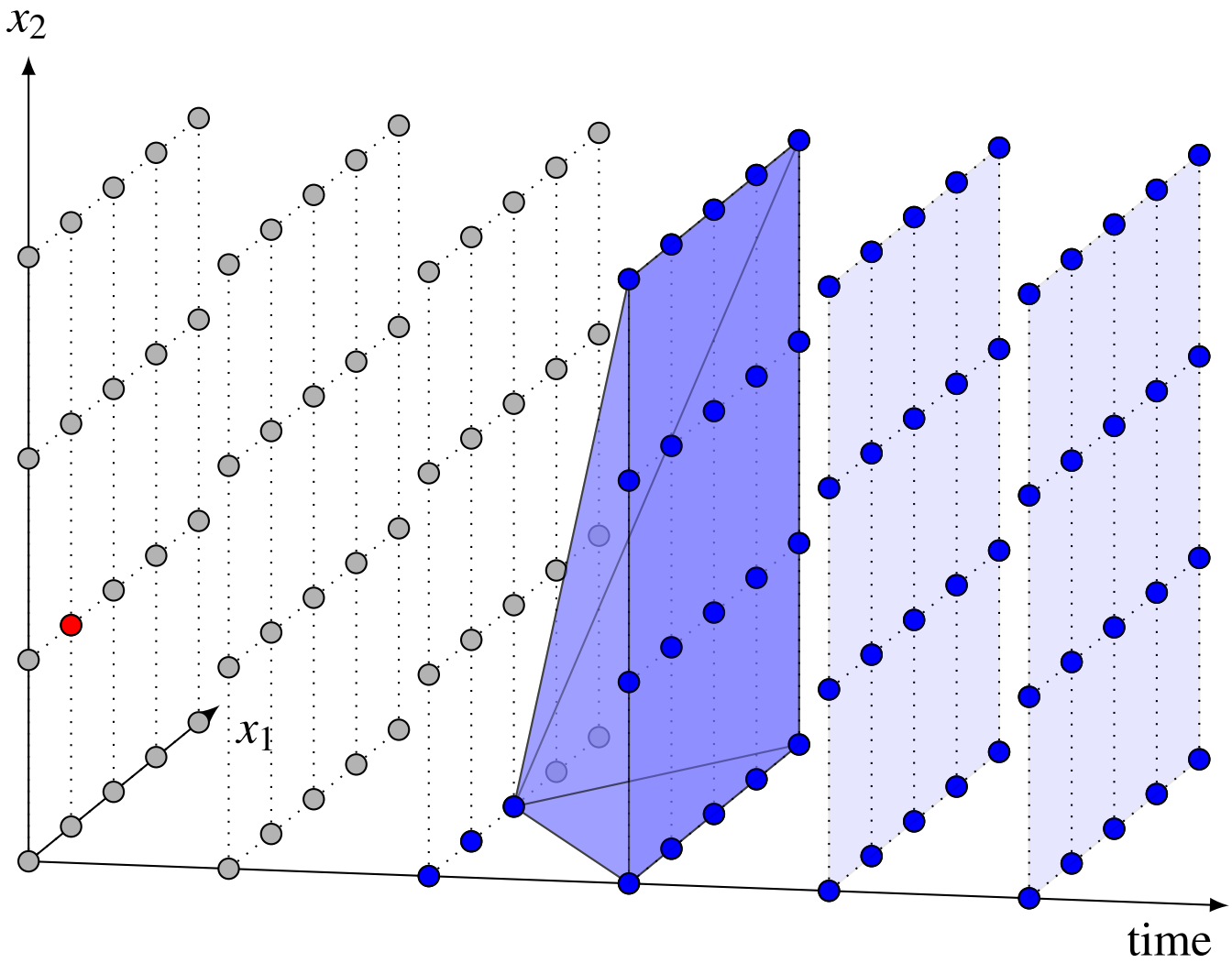
# Dynamic Programming: Illustration



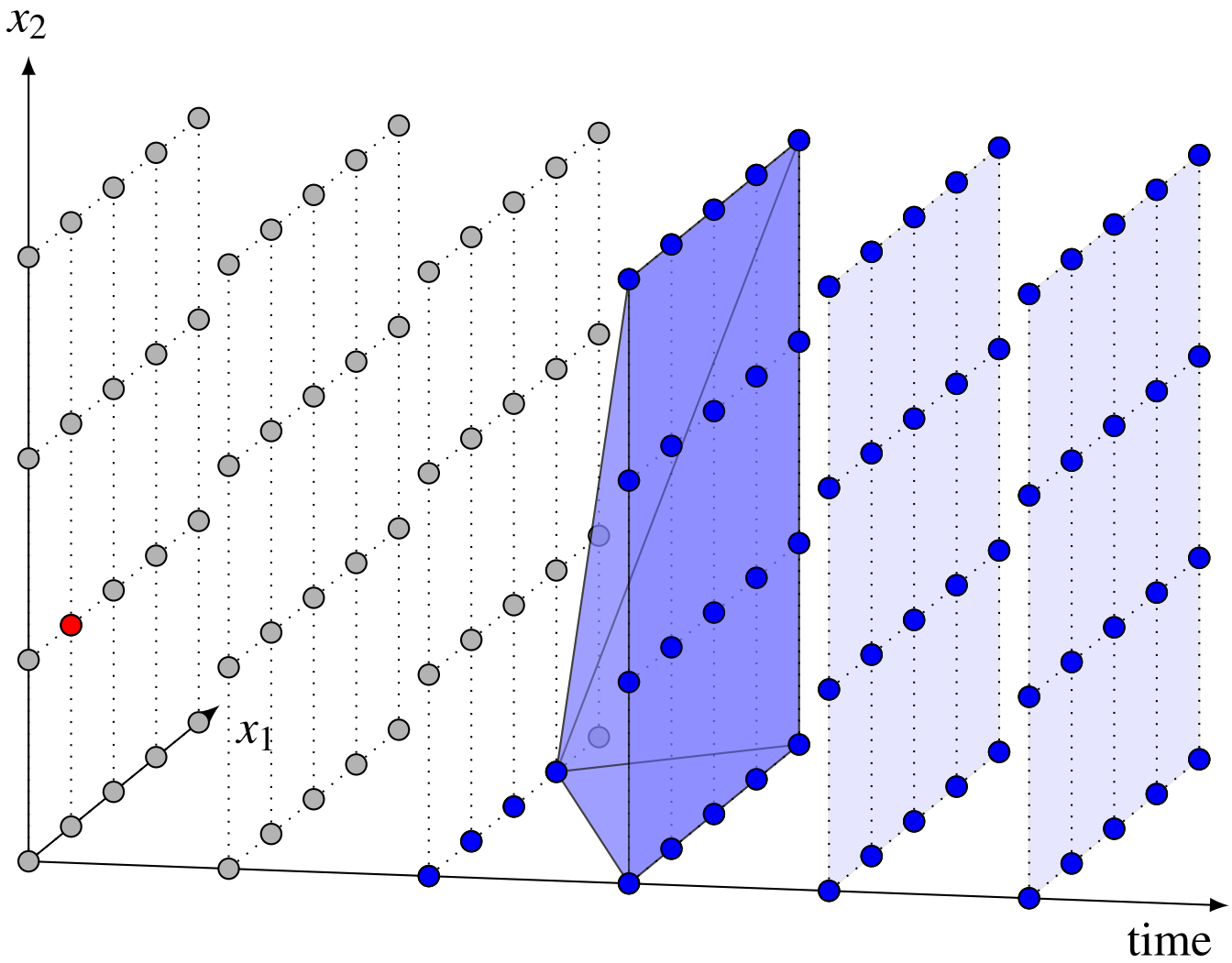
# Dynamic Programming: Illustration



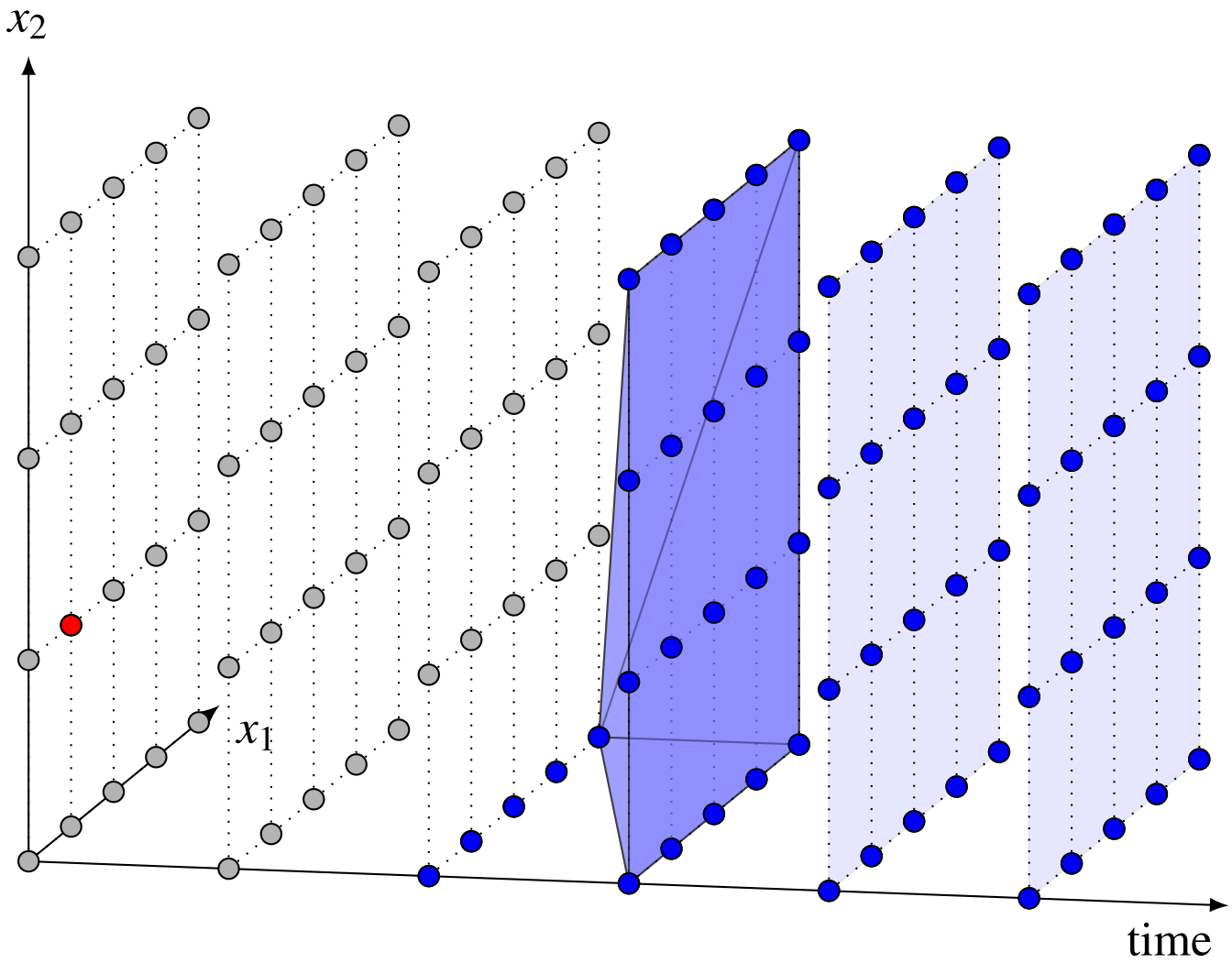
# Dynamic Programming: Illustration



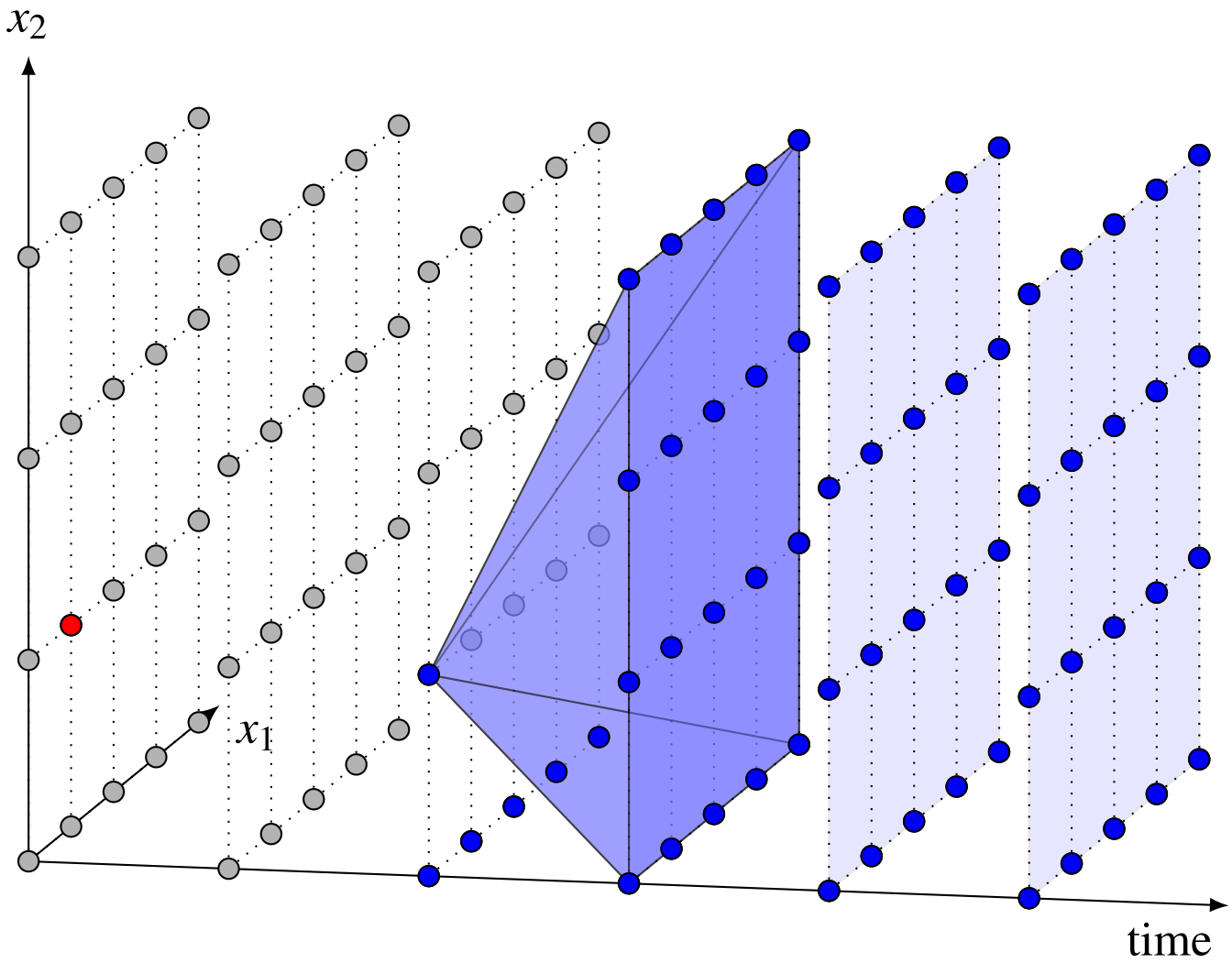
# Dynamic Programming: Illustration



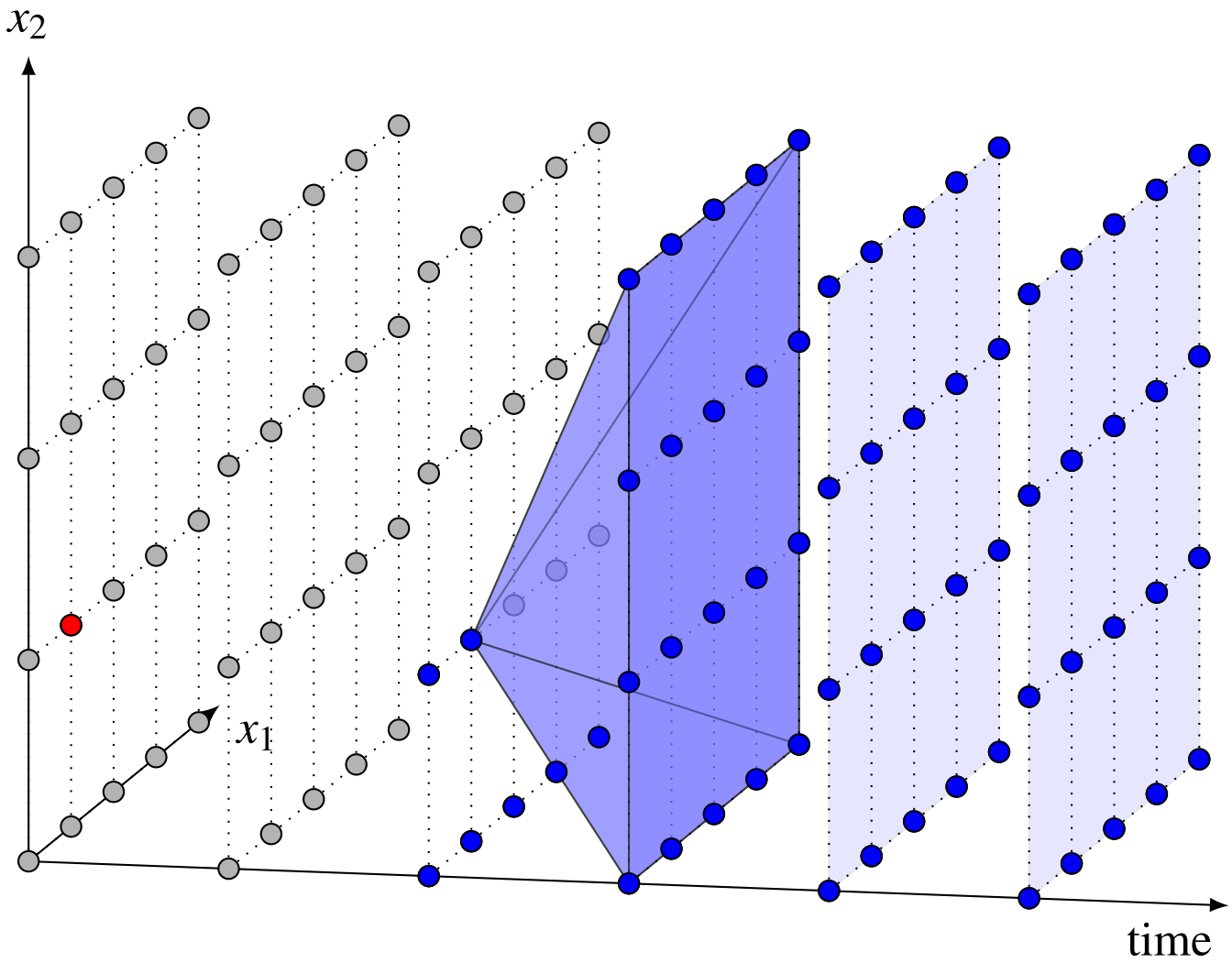
# Dynamic Programming: Illustration



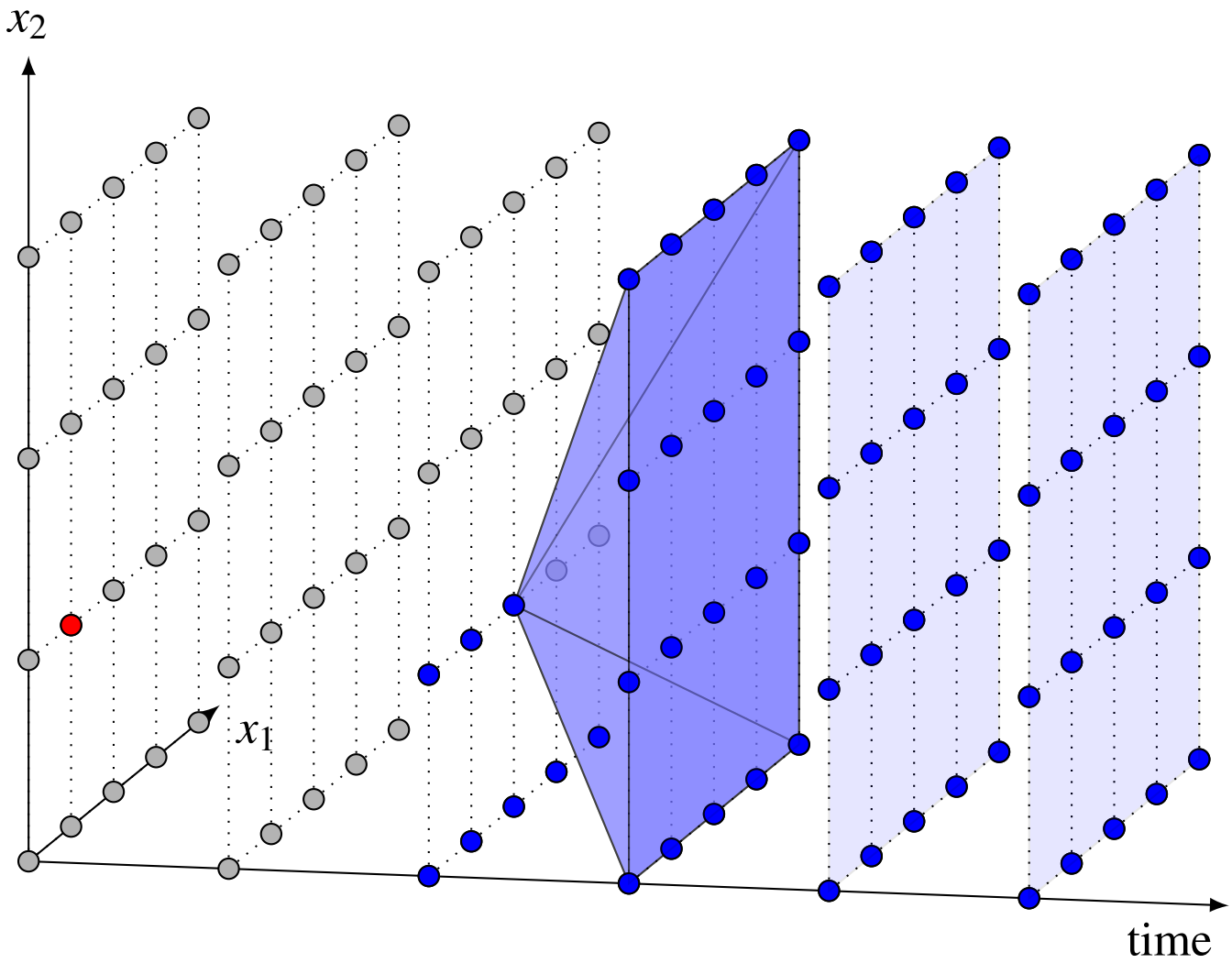
# Dynamic Programming: Illustration



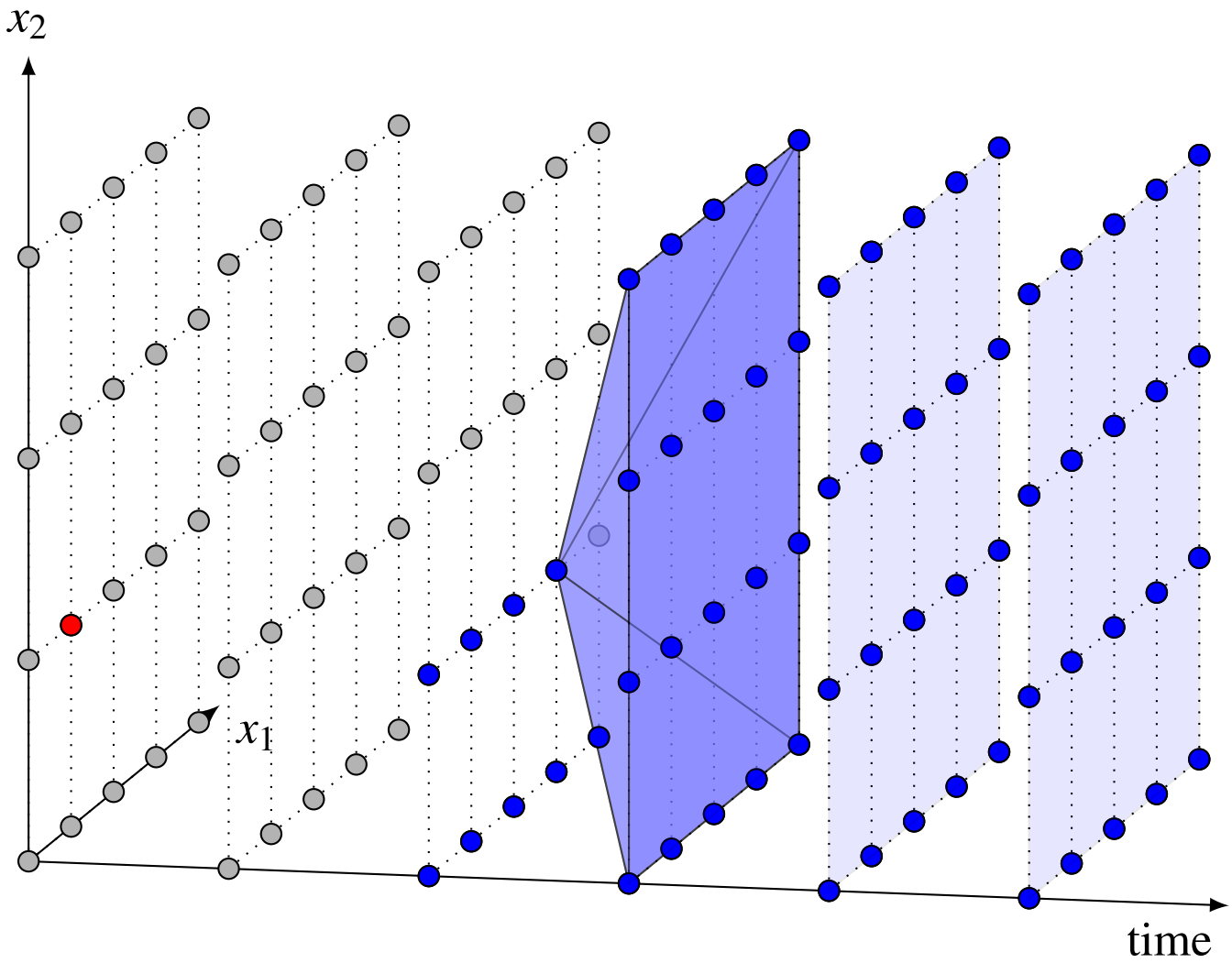
# Dynamic Programming: Illustration



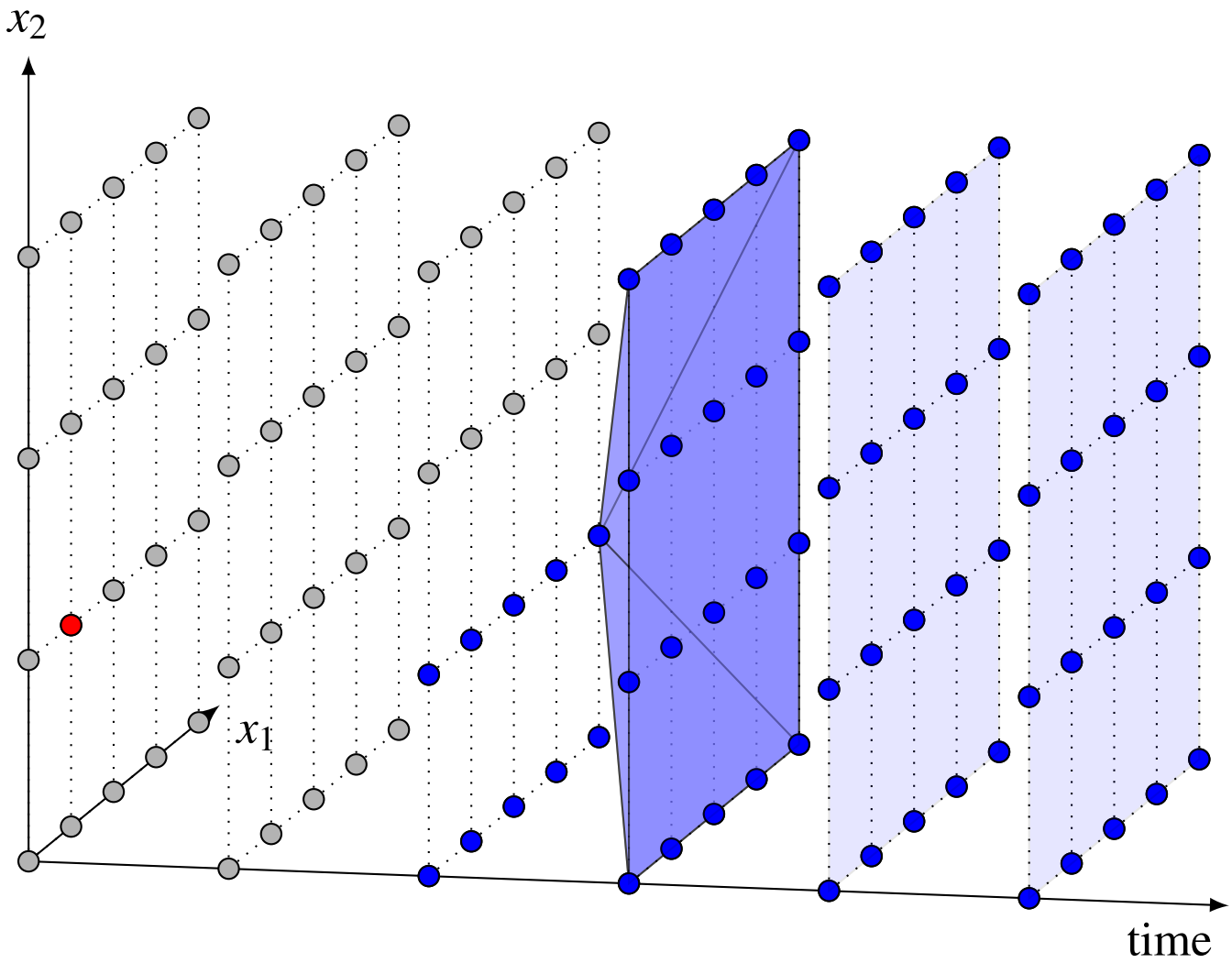
# Dynamic Programming: Illustration



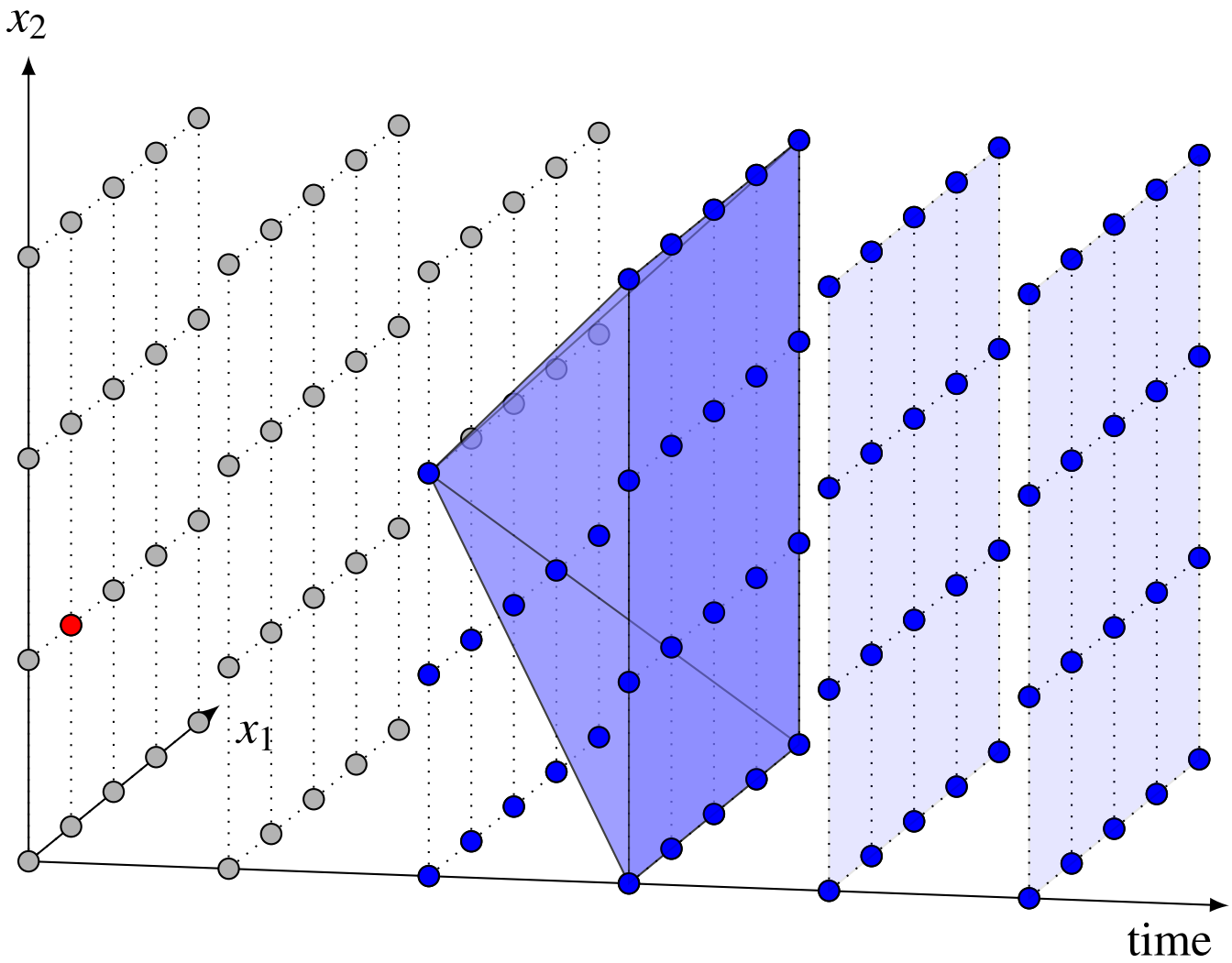
# Dynamic Programming: Illustration



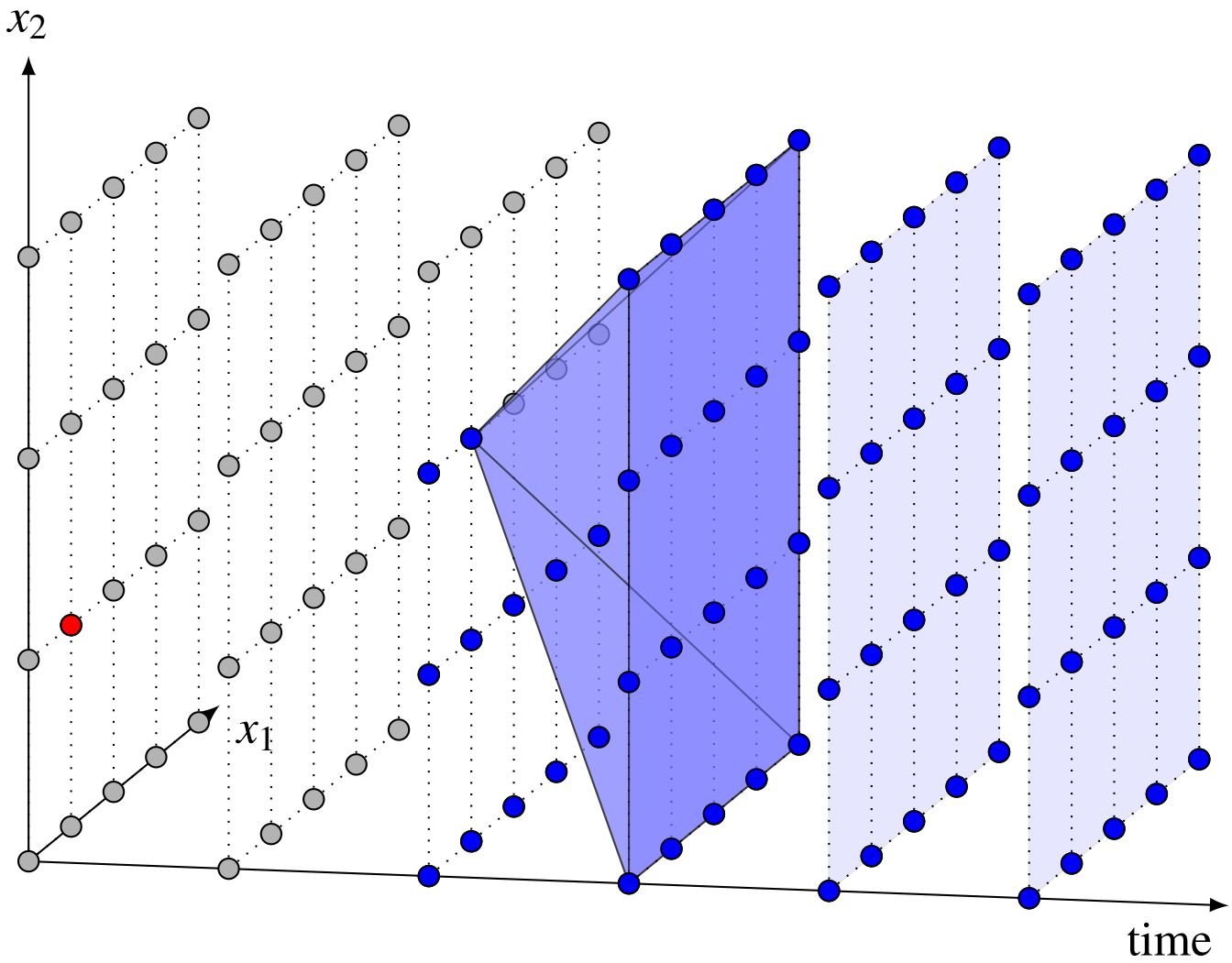
# Dynamic Programming: Illustration



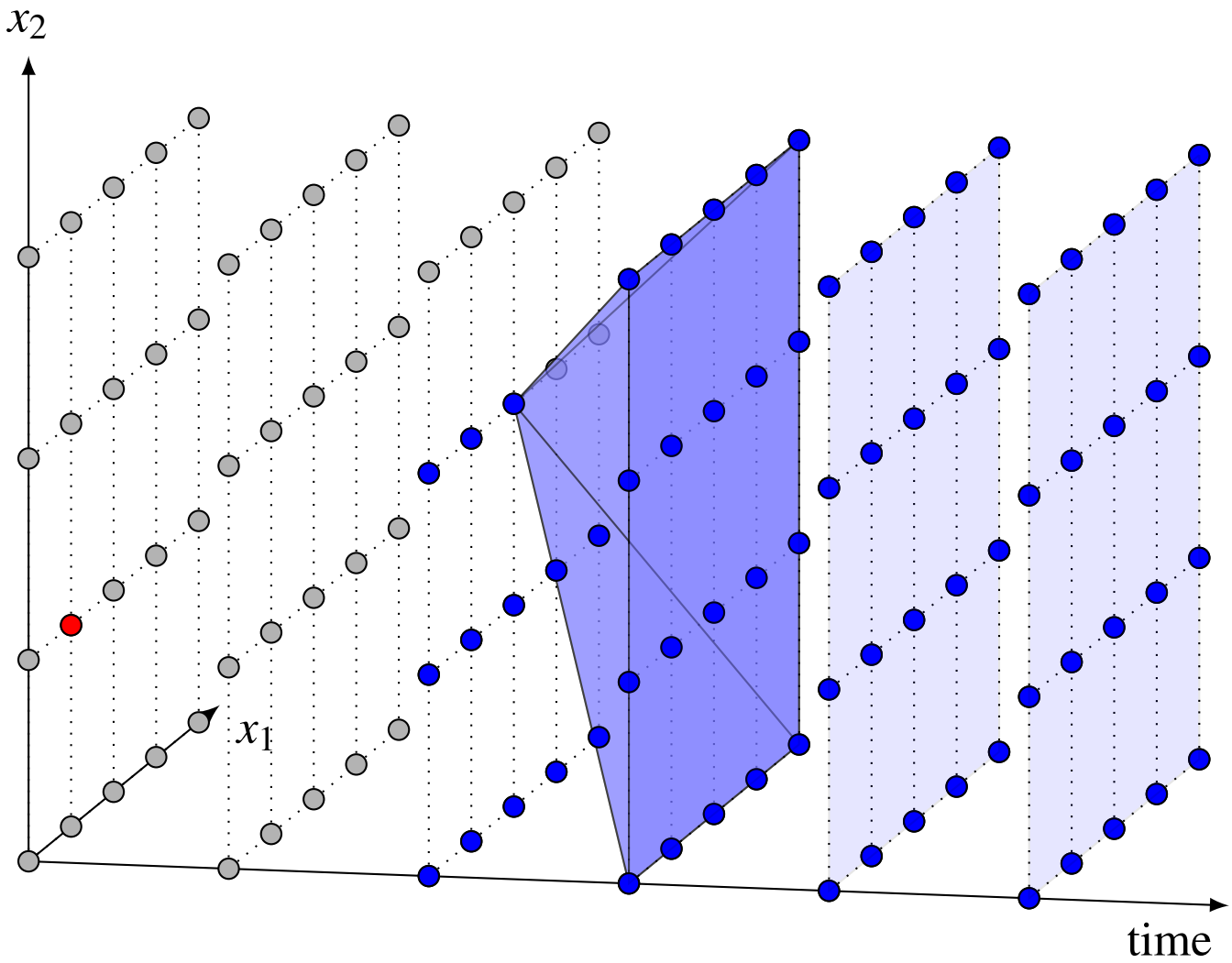
# Dynamic Programming: Illustration



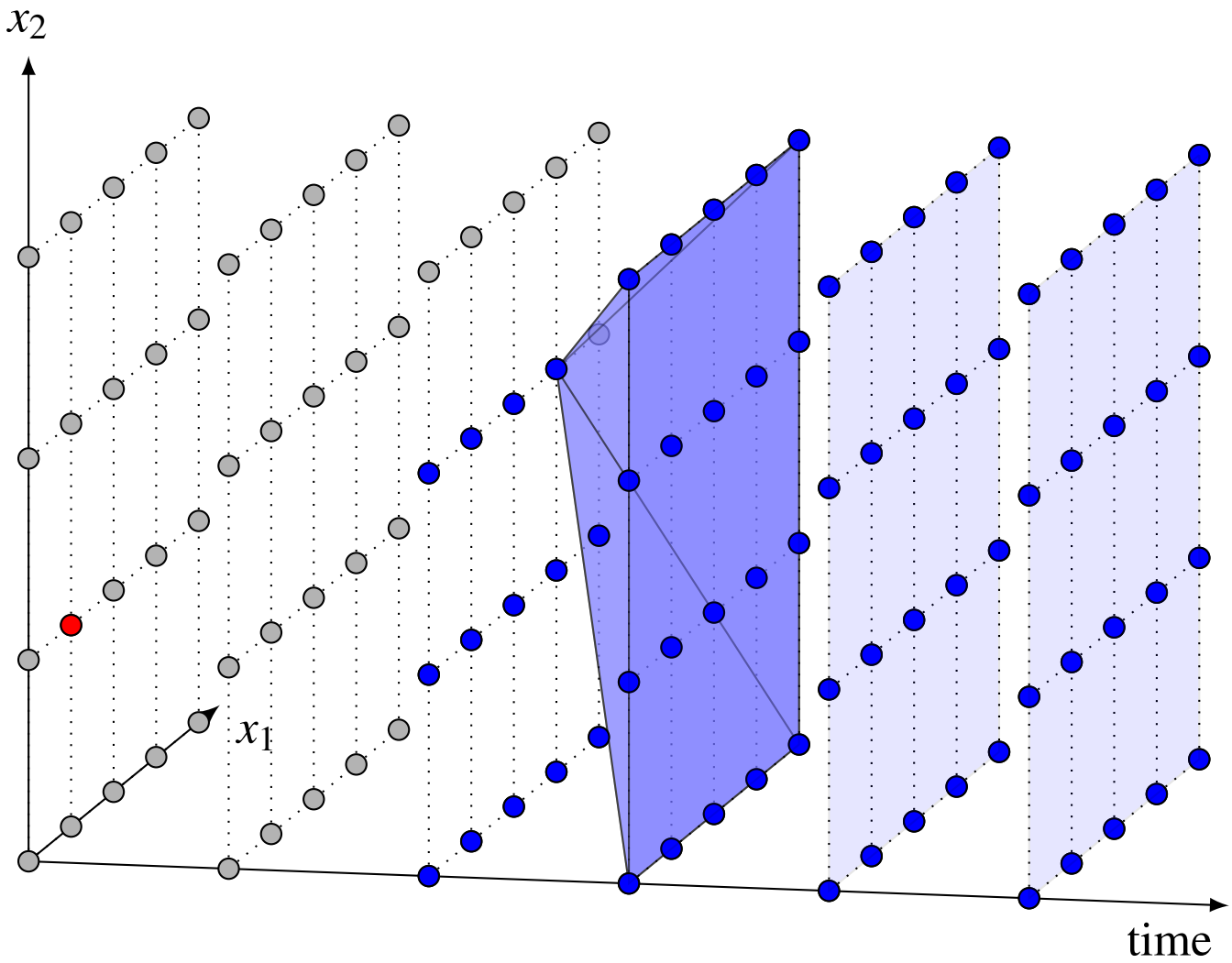
# Dynamic Programming: Illustration



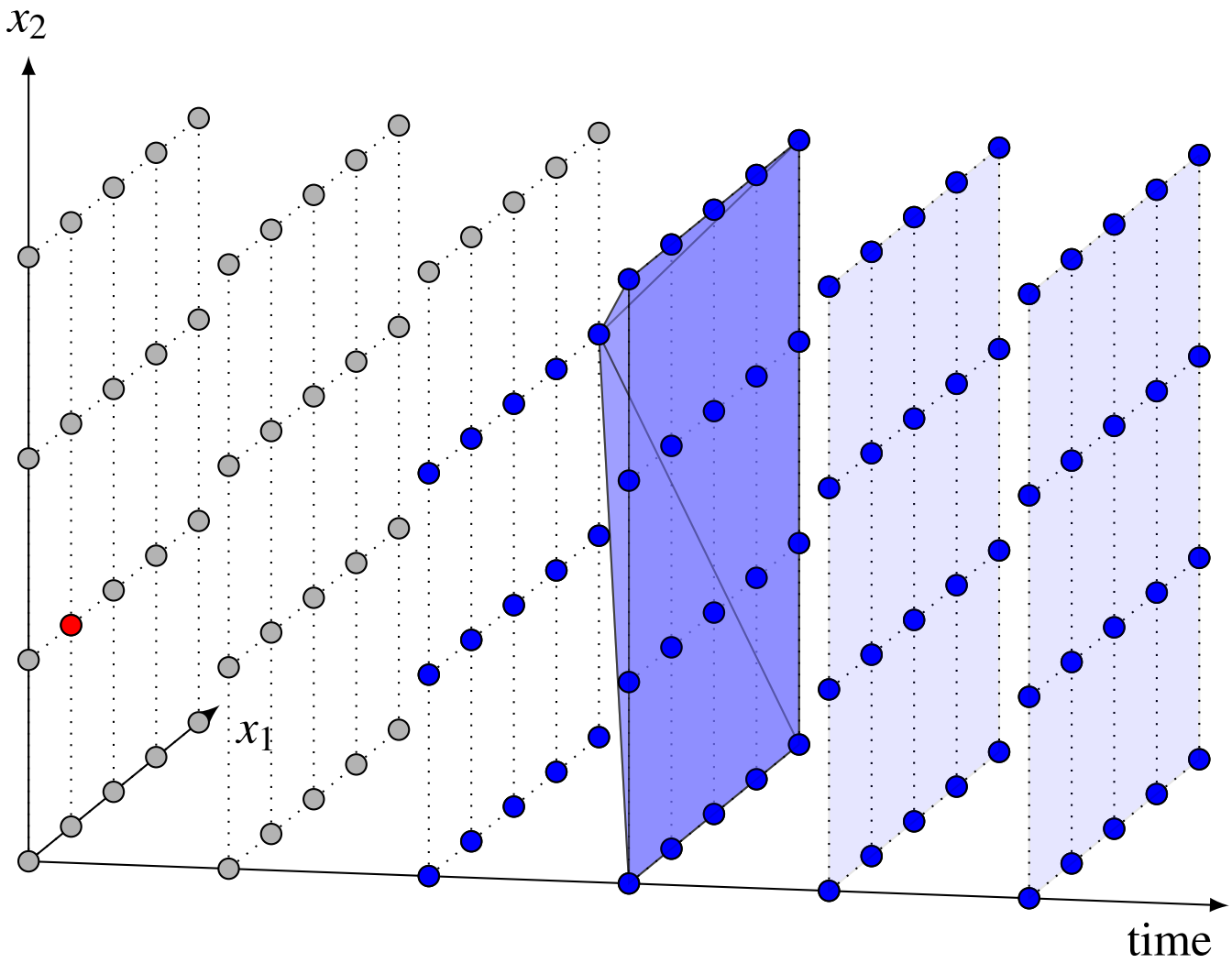
# Dynamic Programming: Illustration



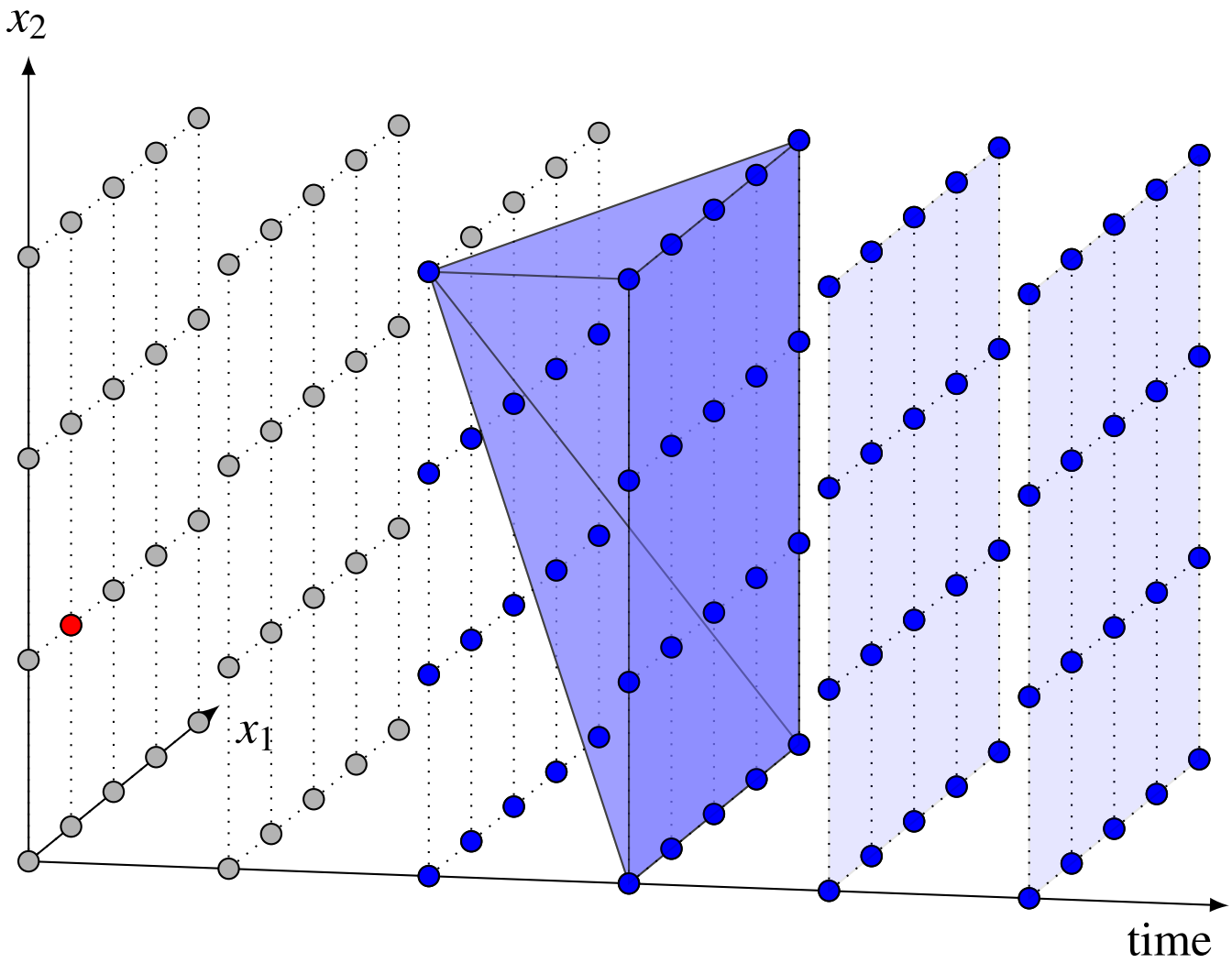
# Dynamic Programming: Illustration



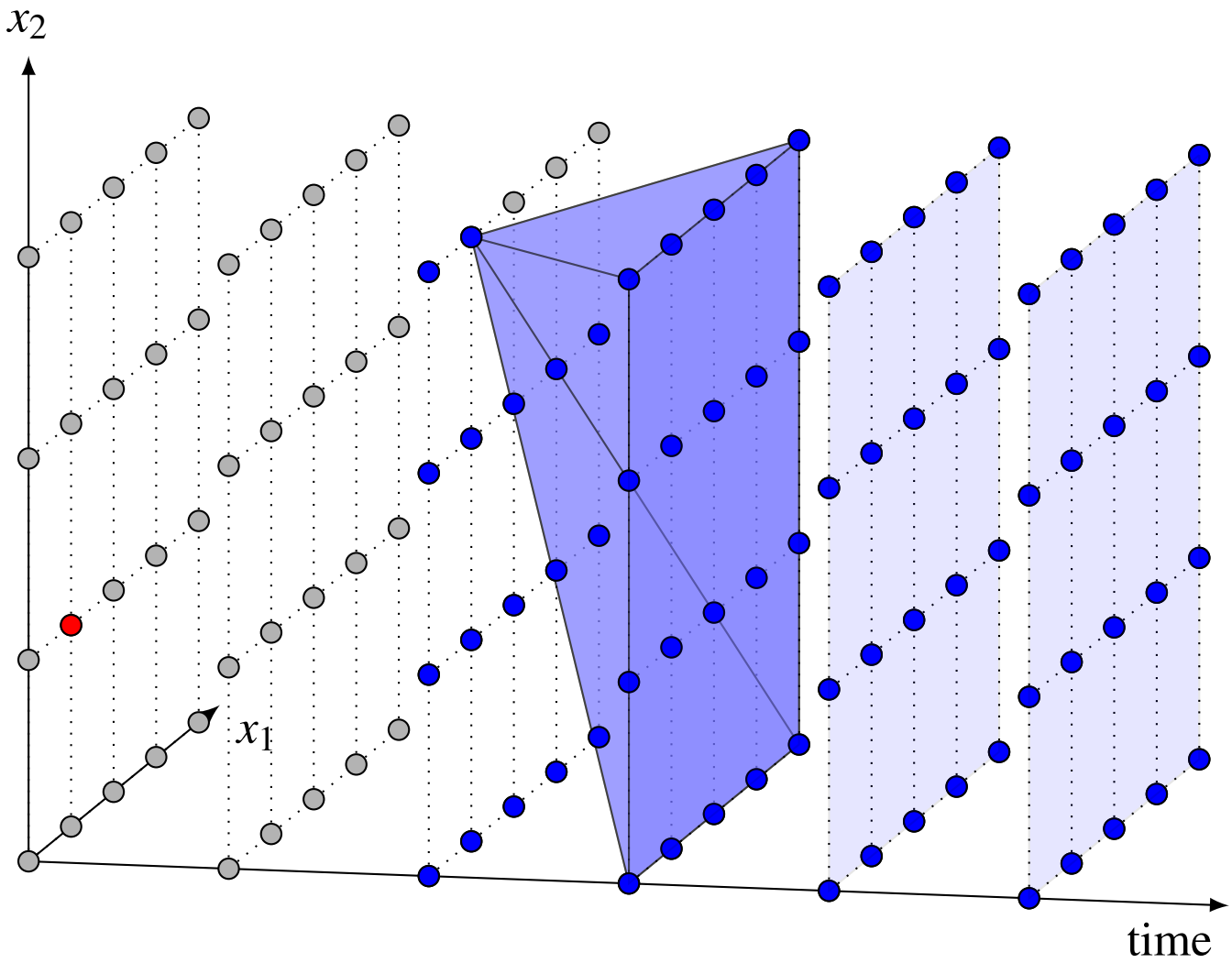
# Dynamic Programming: Illustration



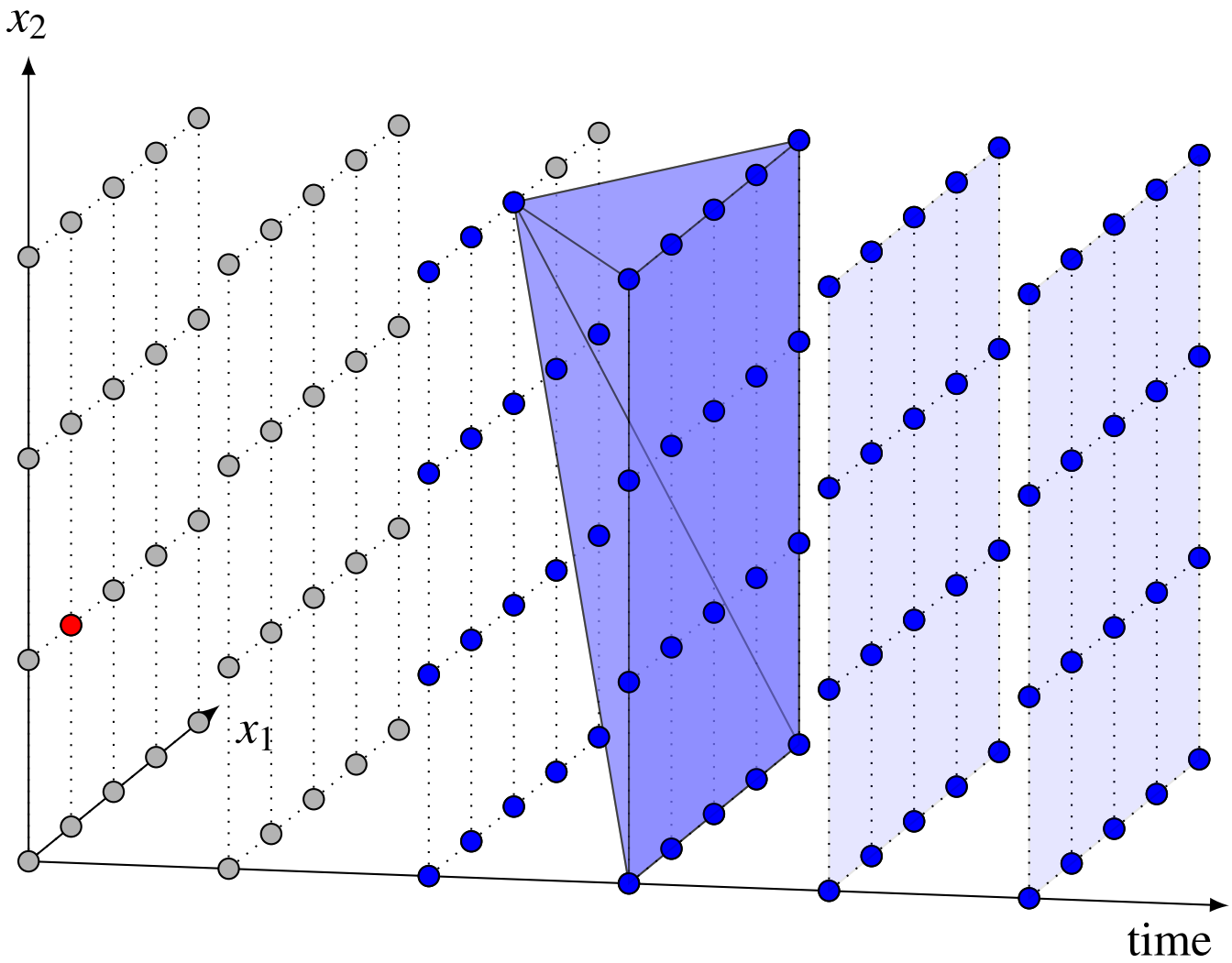
# Dynamic Programming: Illustration



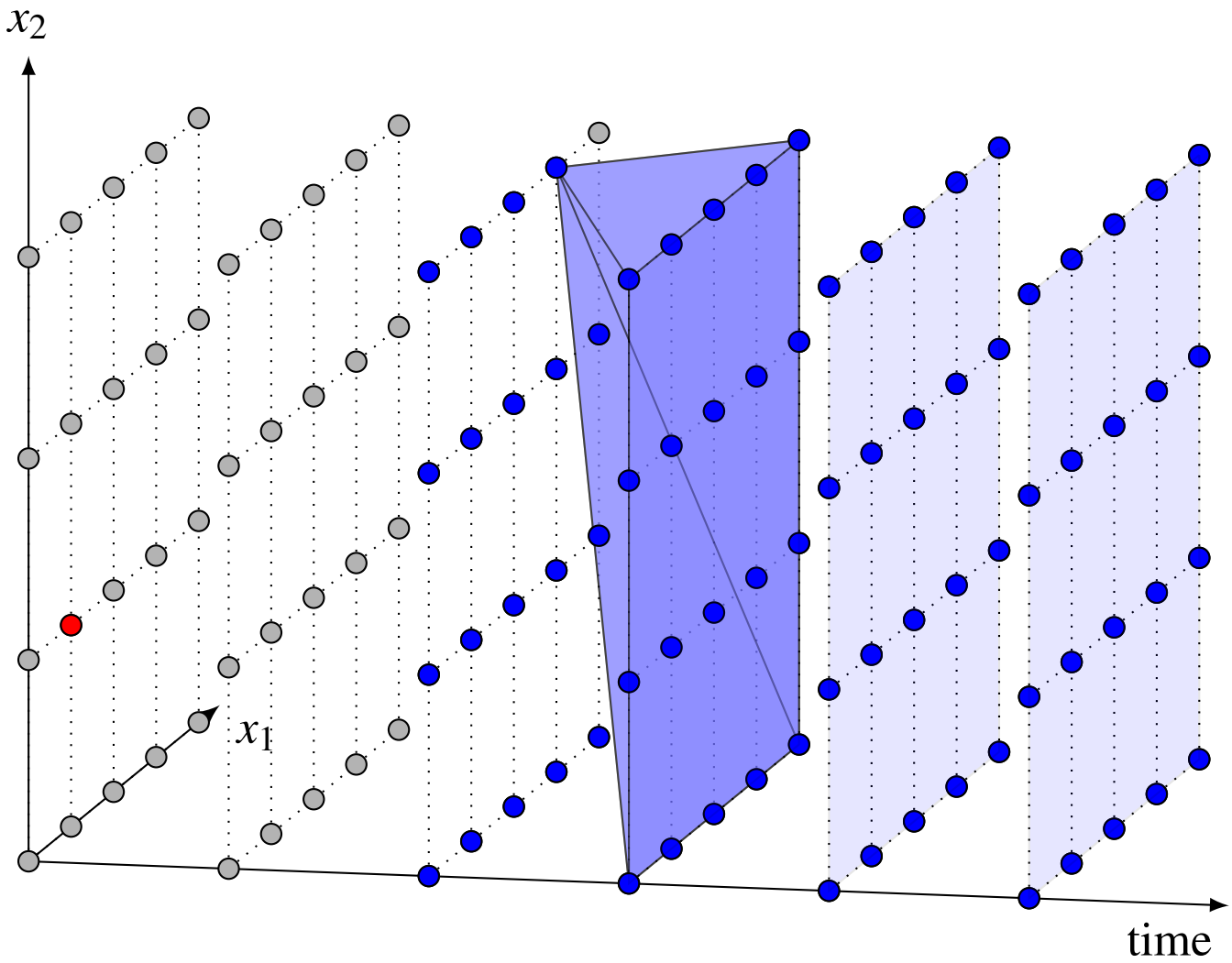
# Dynamic Programming: Illustration



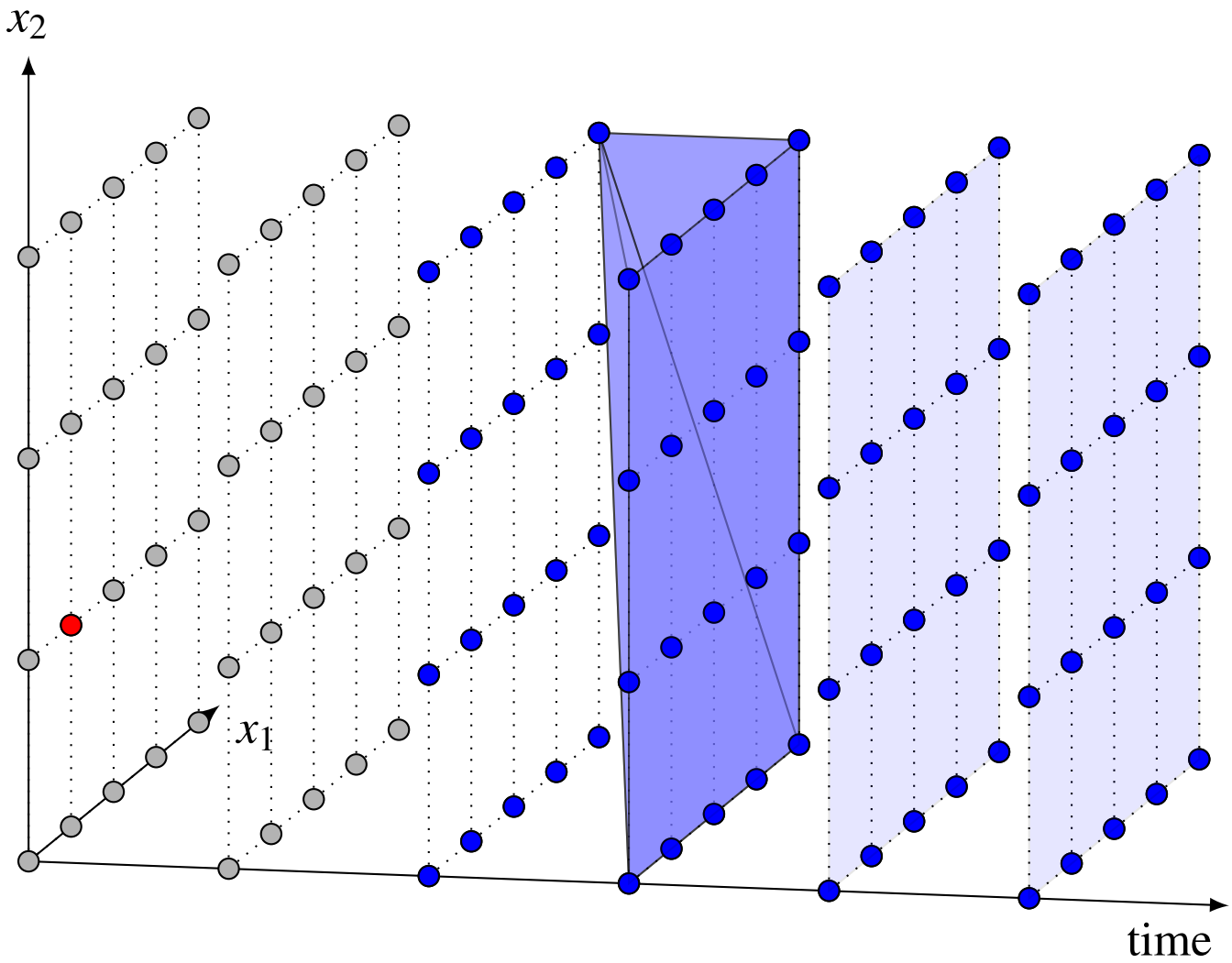
# Dynamic Programming: Illustration



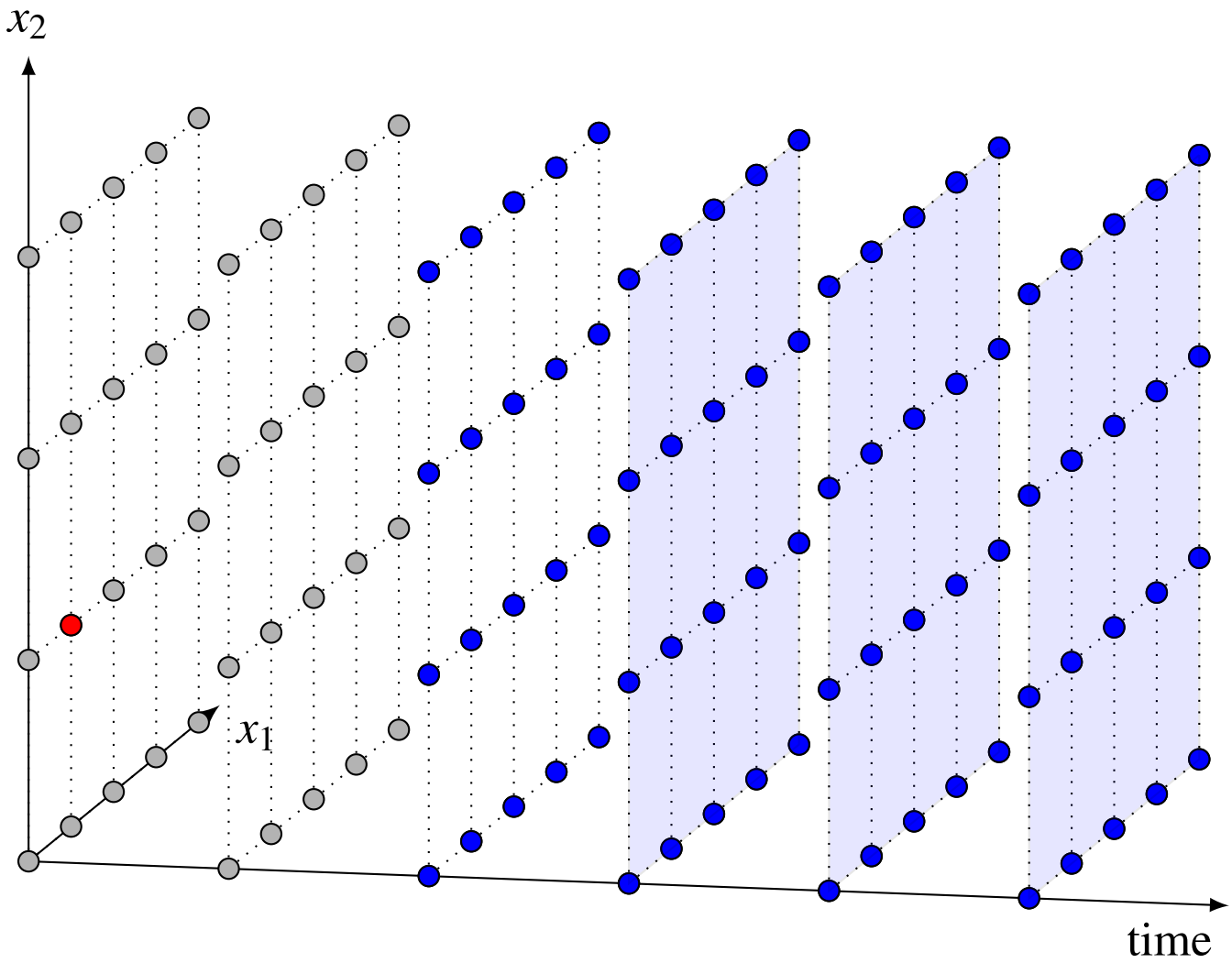
# Dynamic Programming: Illustration



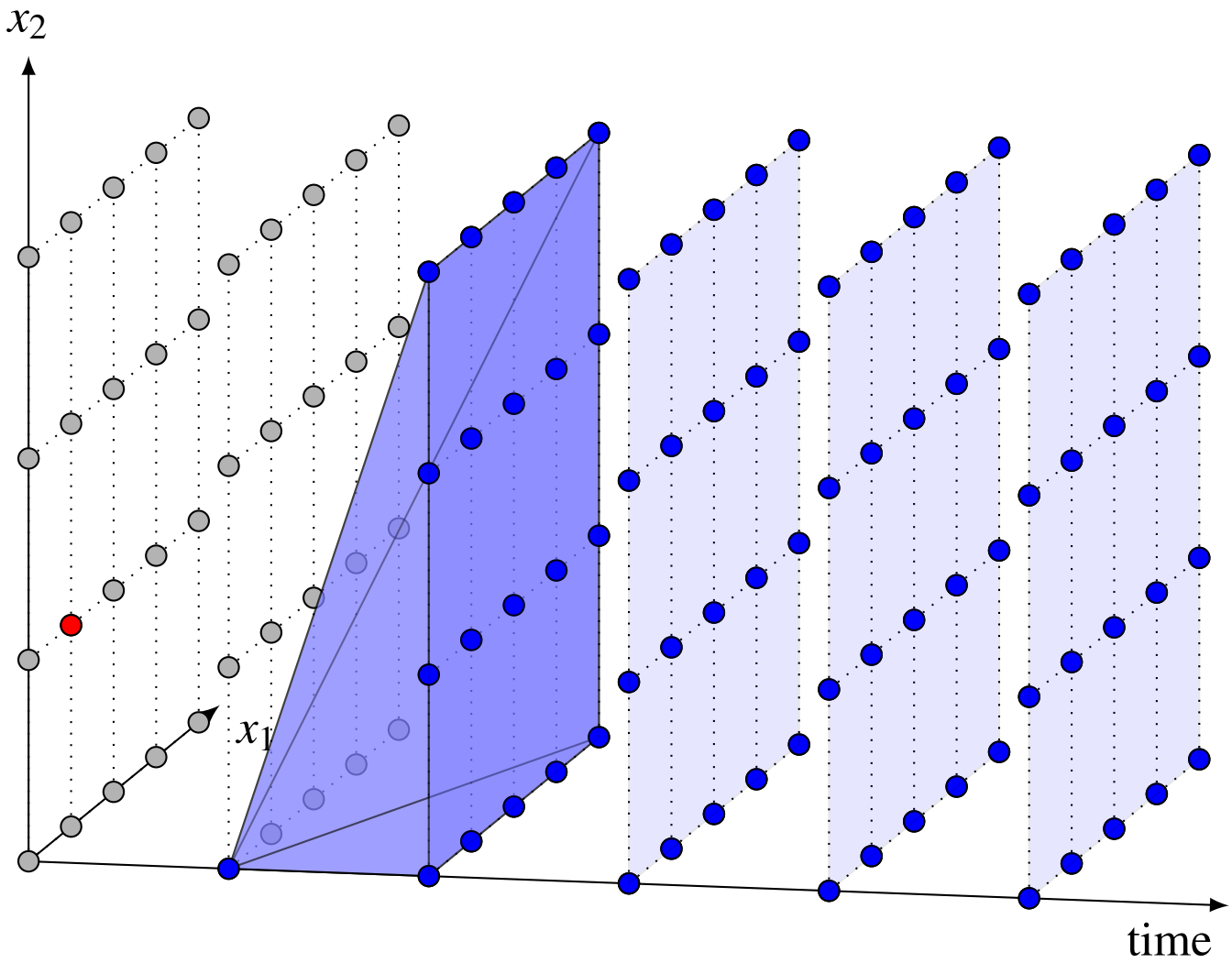
# Dynamic Programming: Illustration



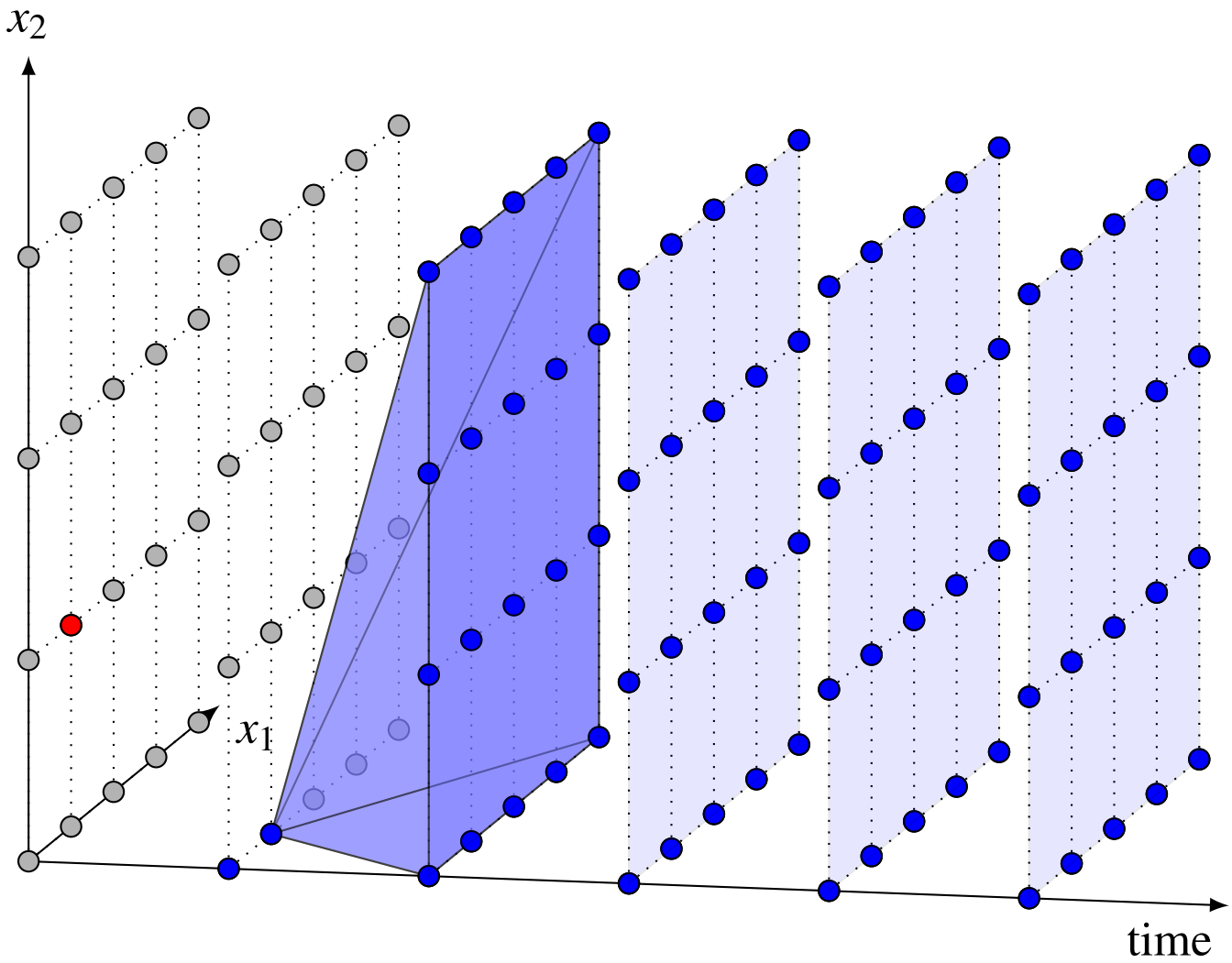
# Dynamic Programming: Illustration



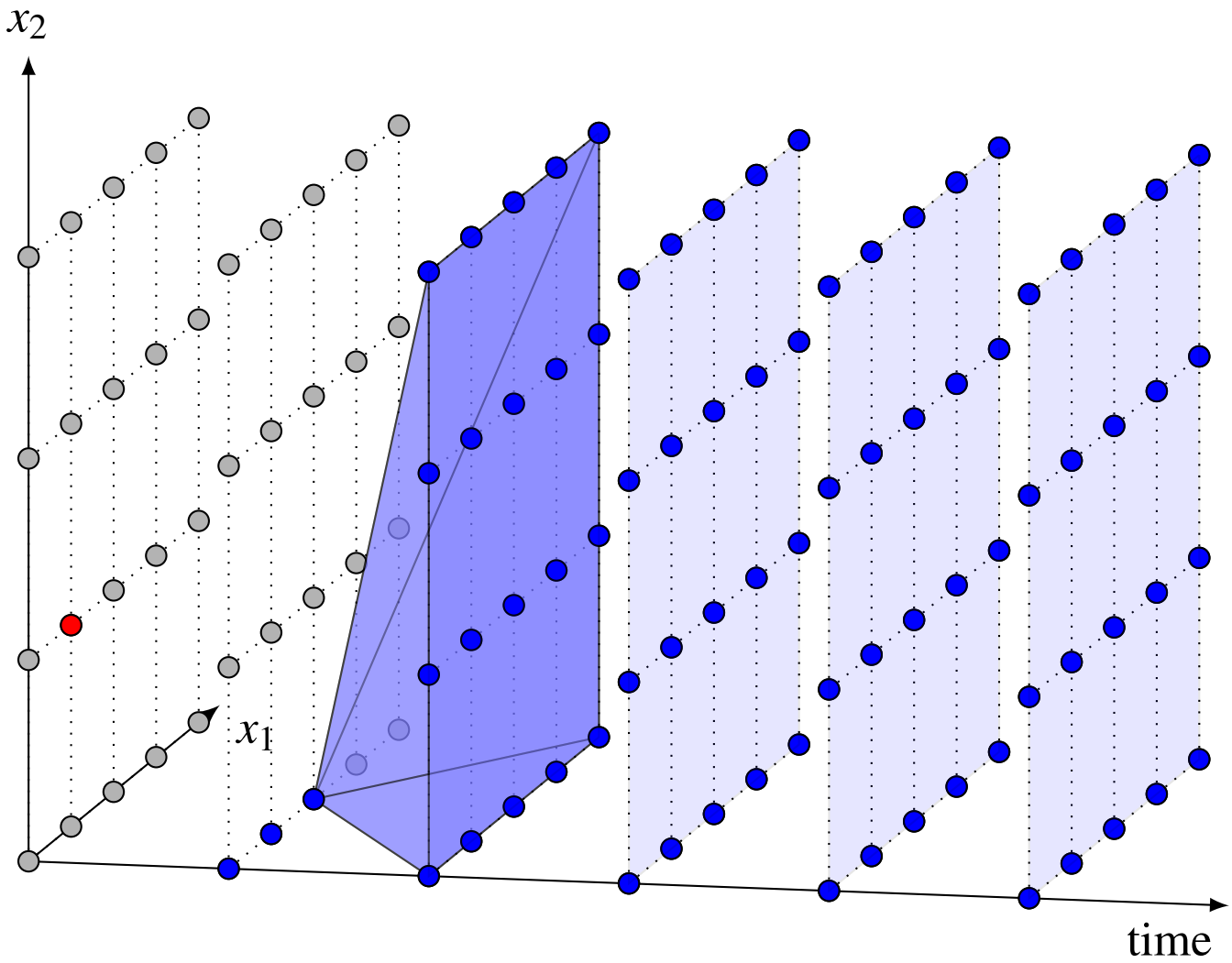
# Dynamic Programming: Illustration



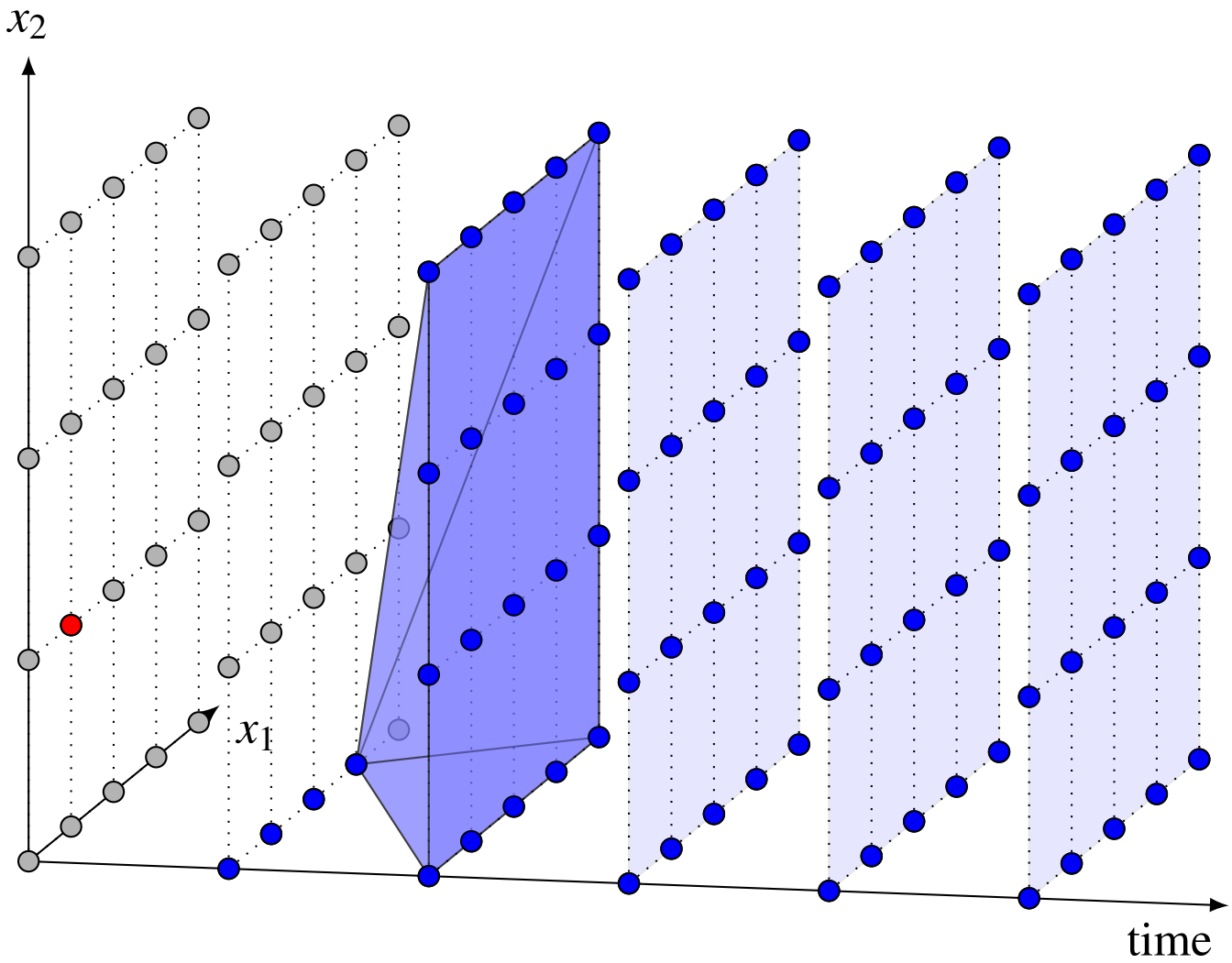
# Dynamic Programming: Illustration



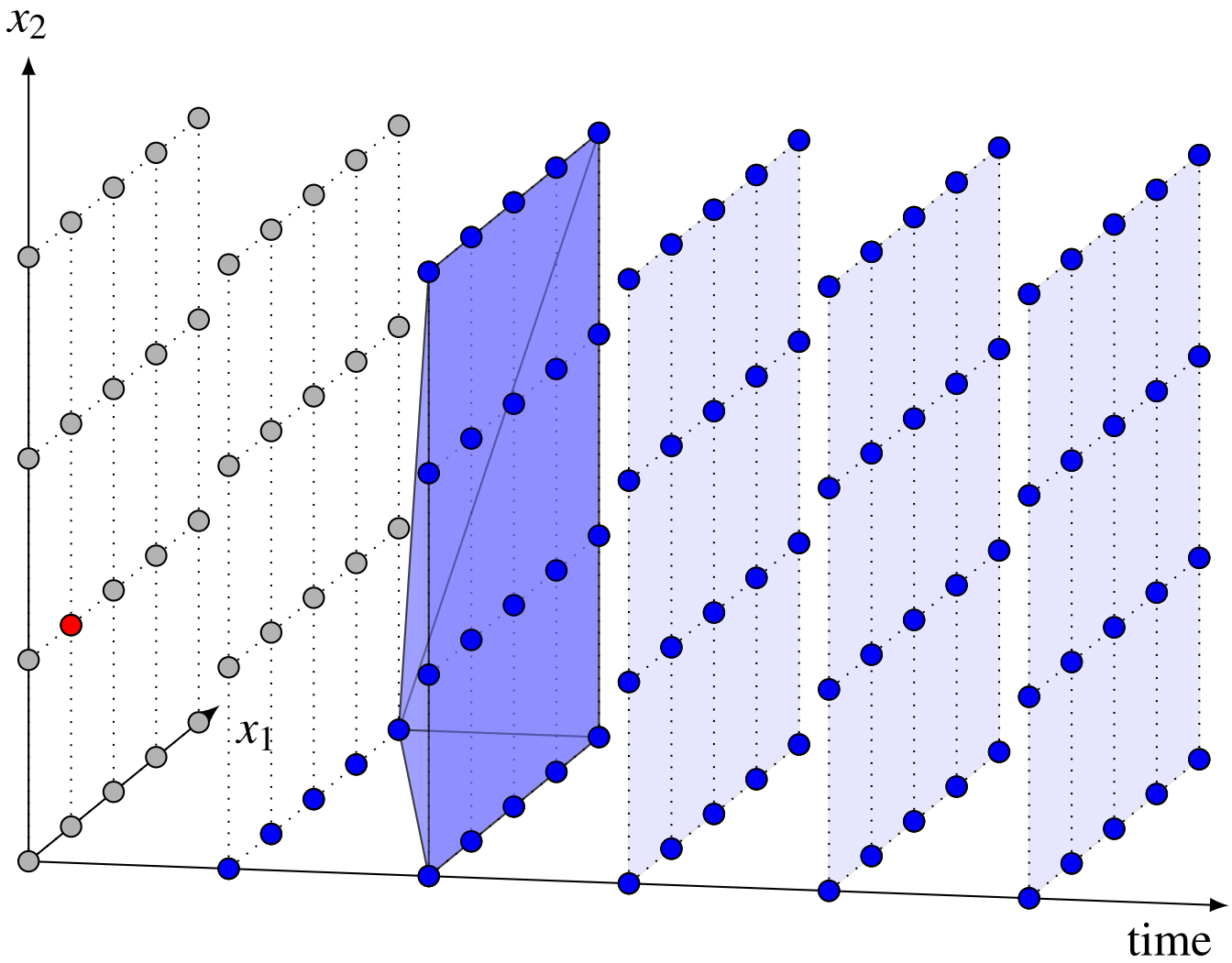
# Dynamic Programming: Illustration



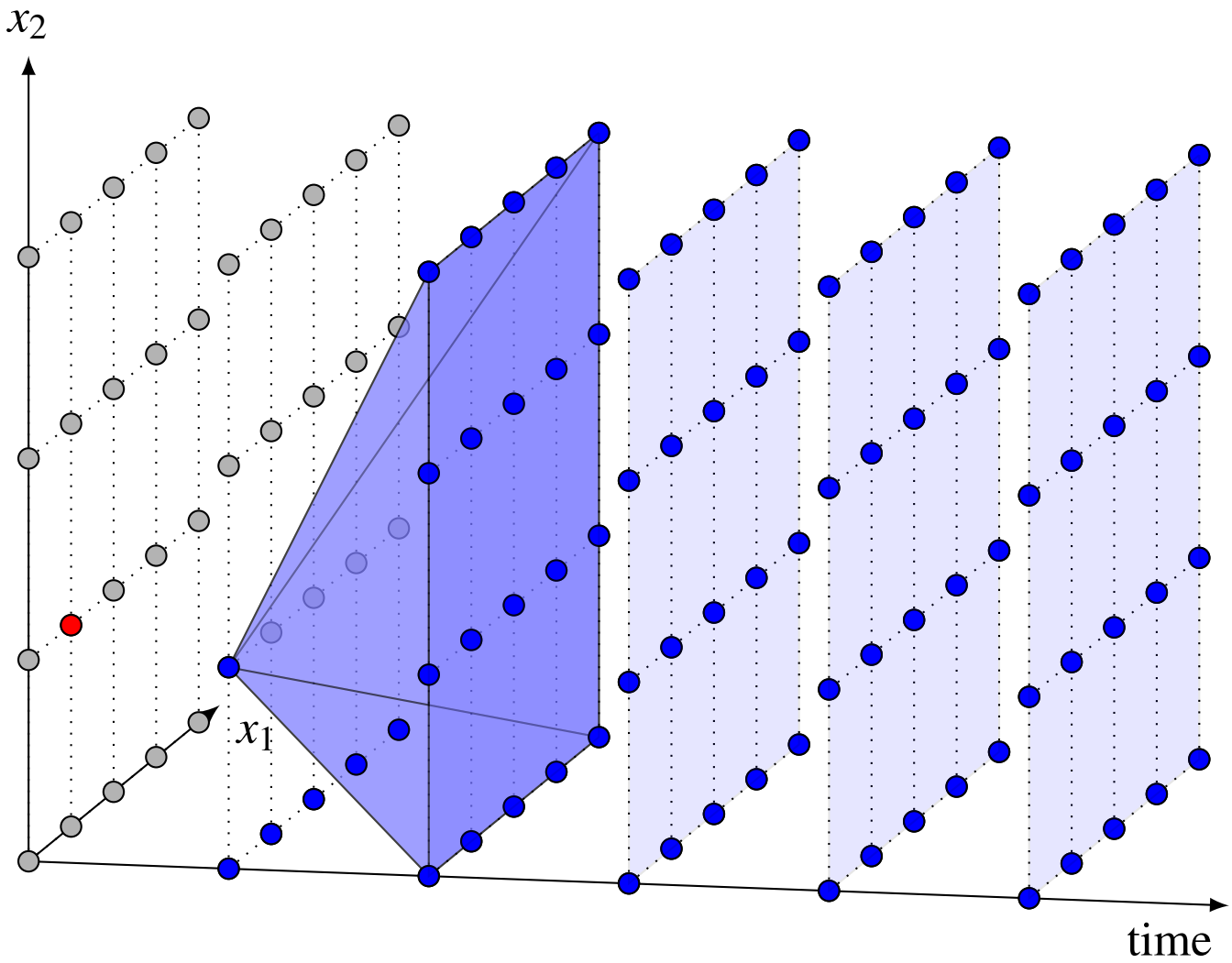
# Dynamic Programming: Illustration



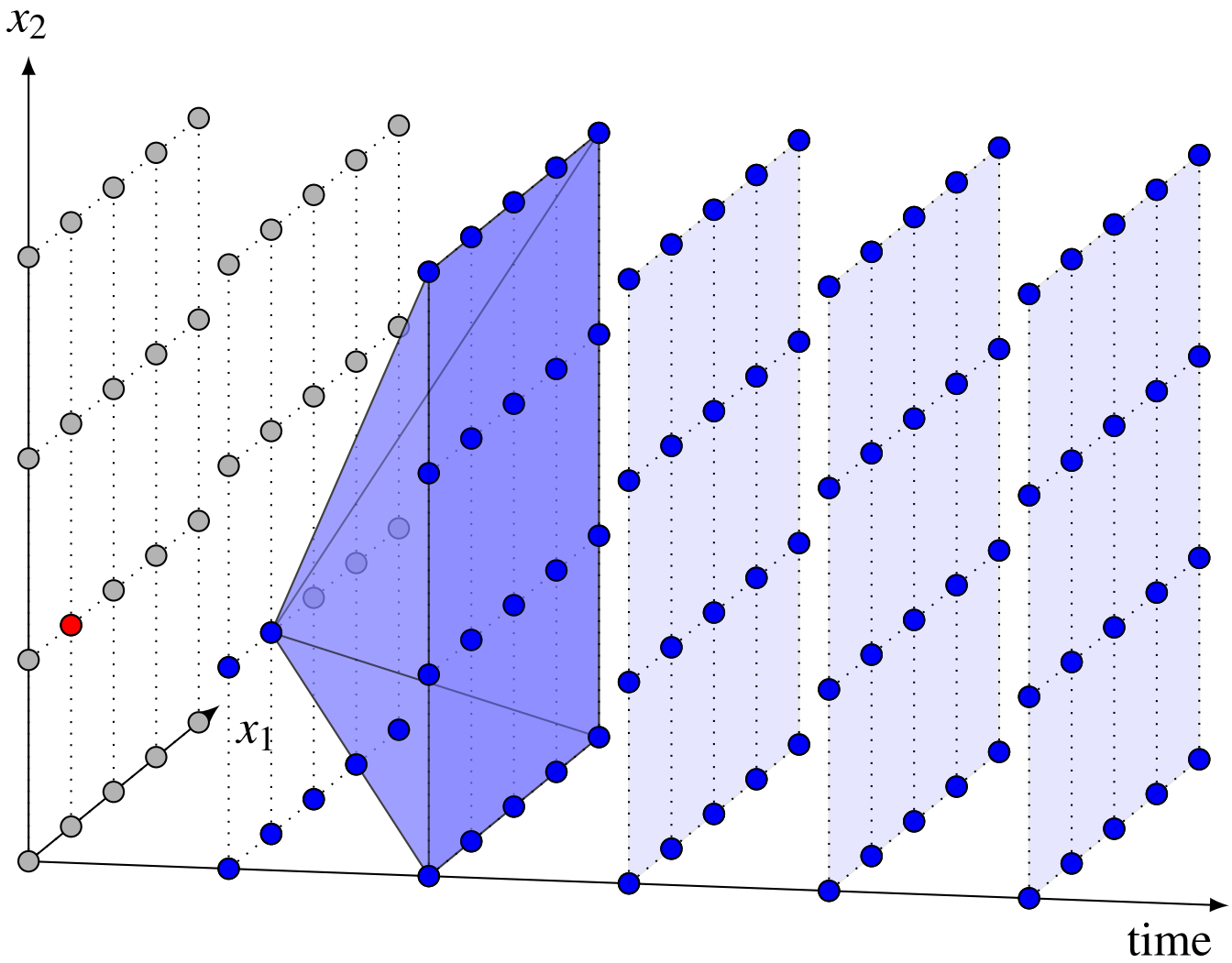
# Dynamic Programming: Illustration



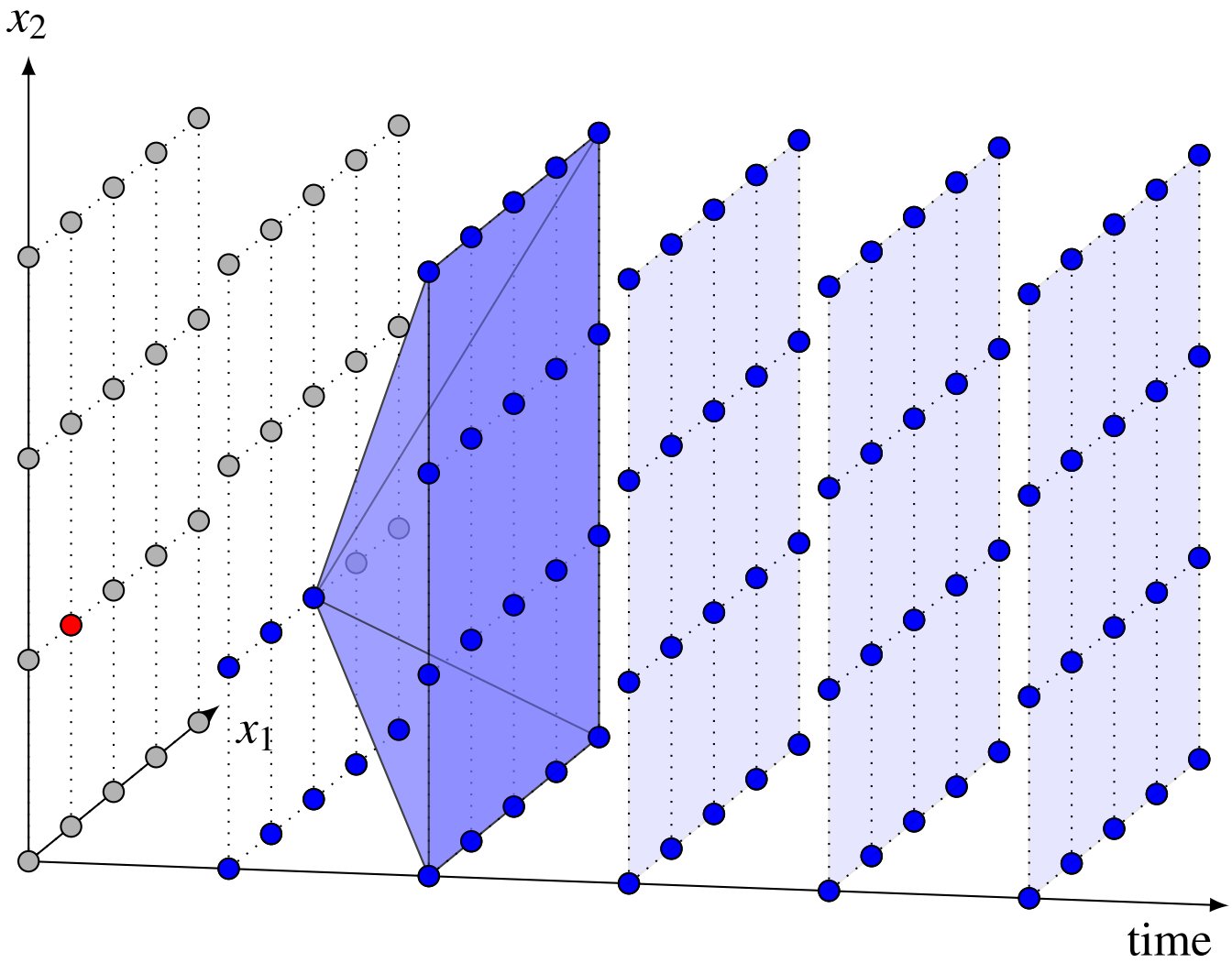
# Dynamic Programming: Illustration



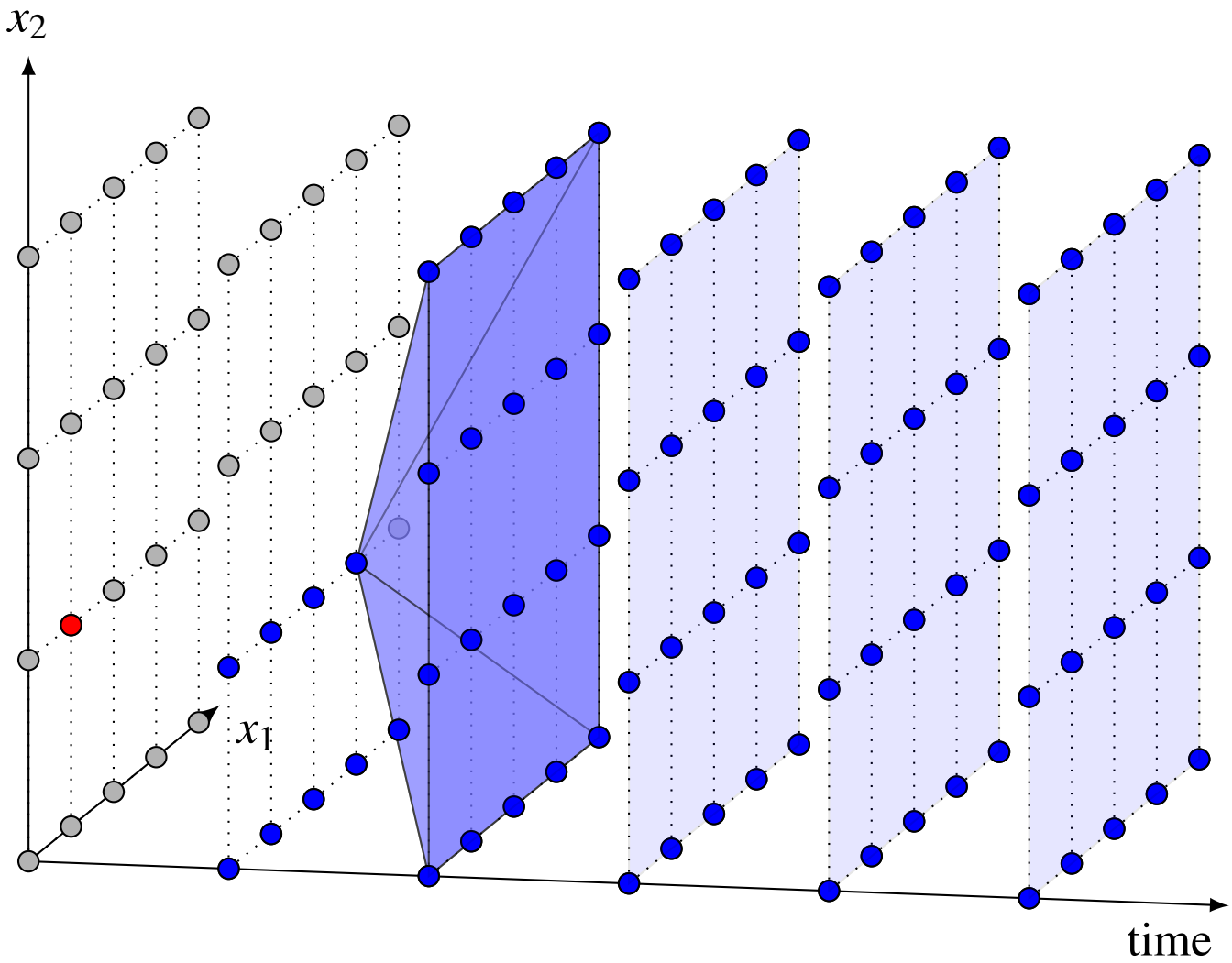
# Dynamic Programming: Illustration



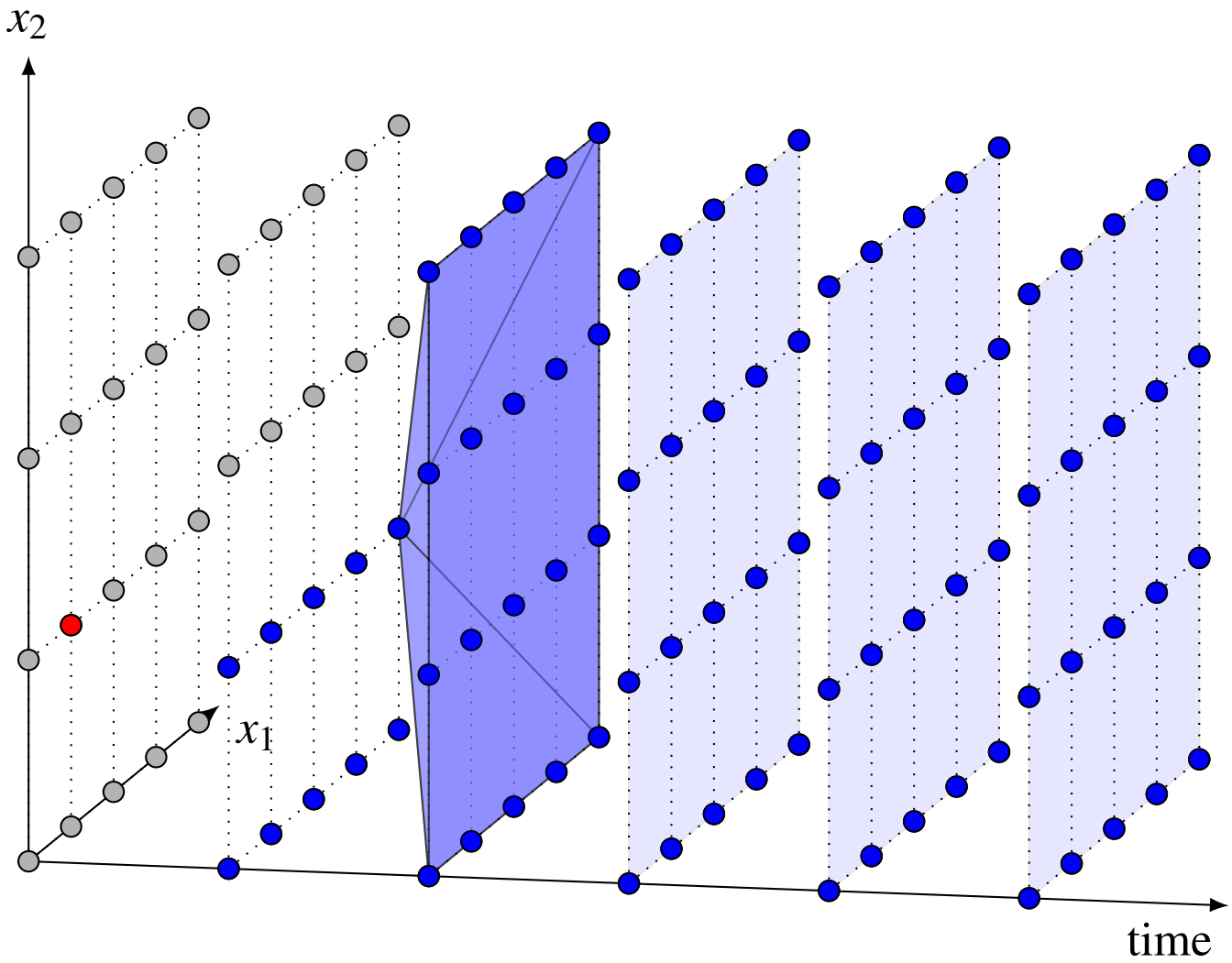
# Dynamic Programming: Illustration



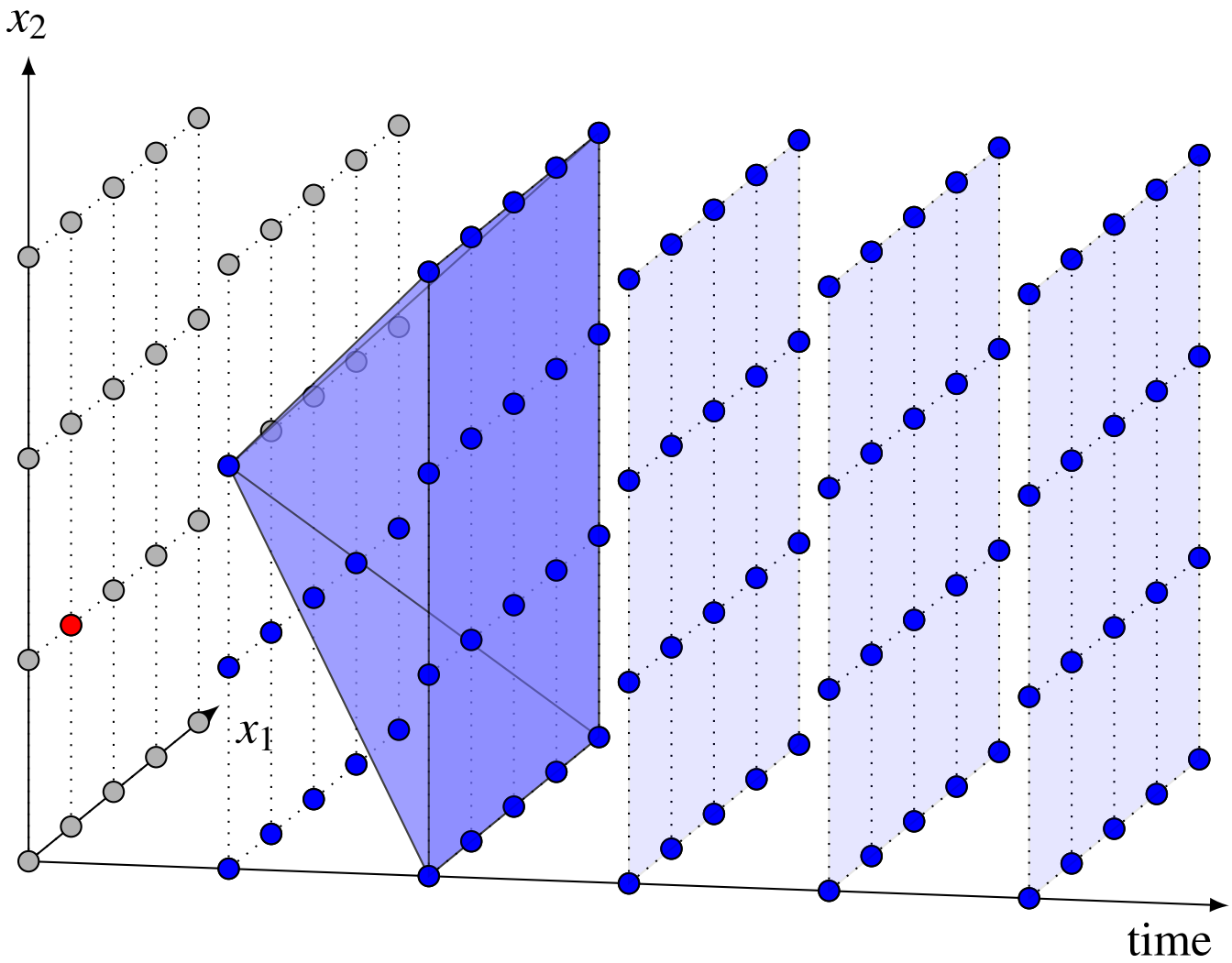
# Dynamic Programming: Illustration



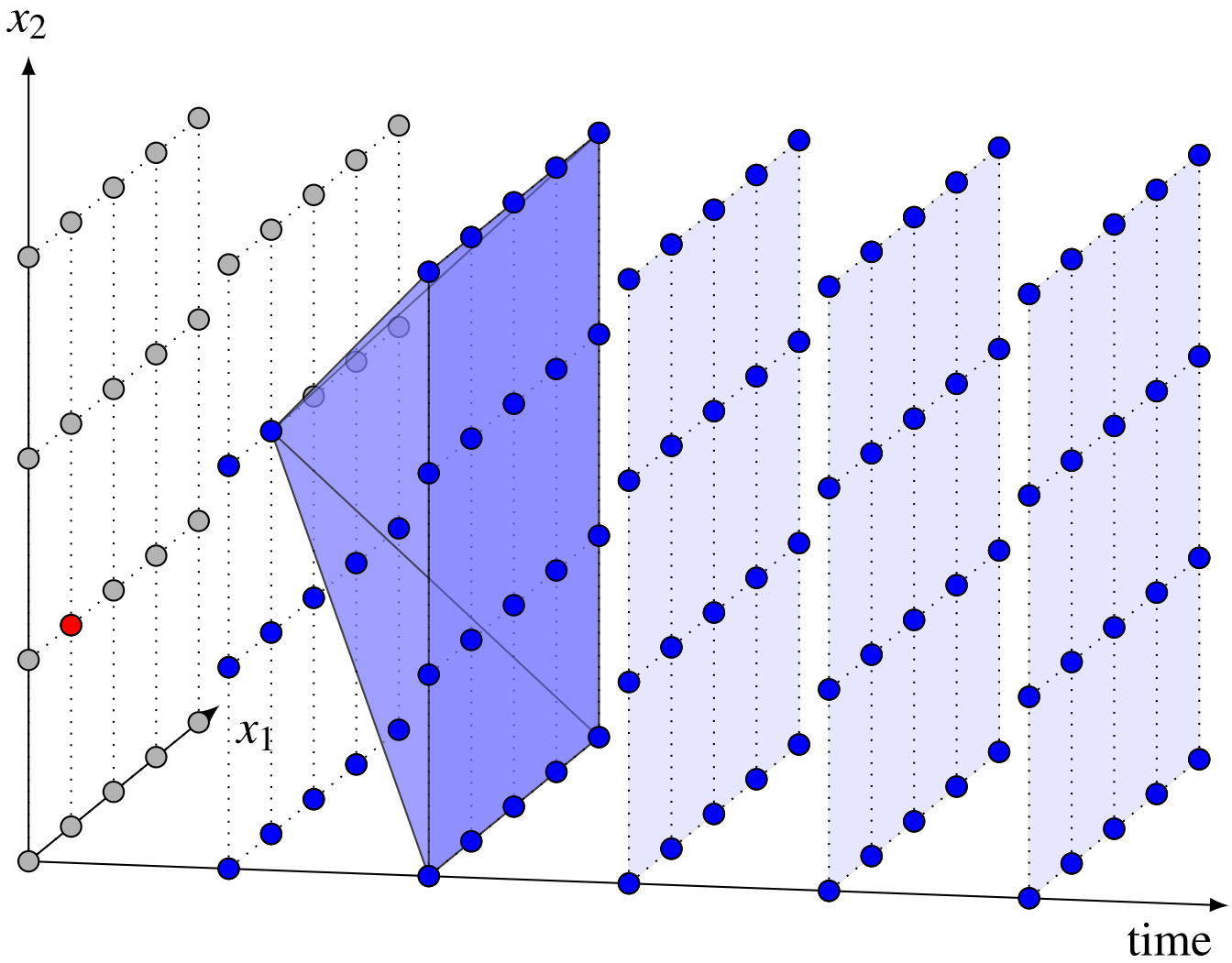
# Dynamic Programming: Illustration



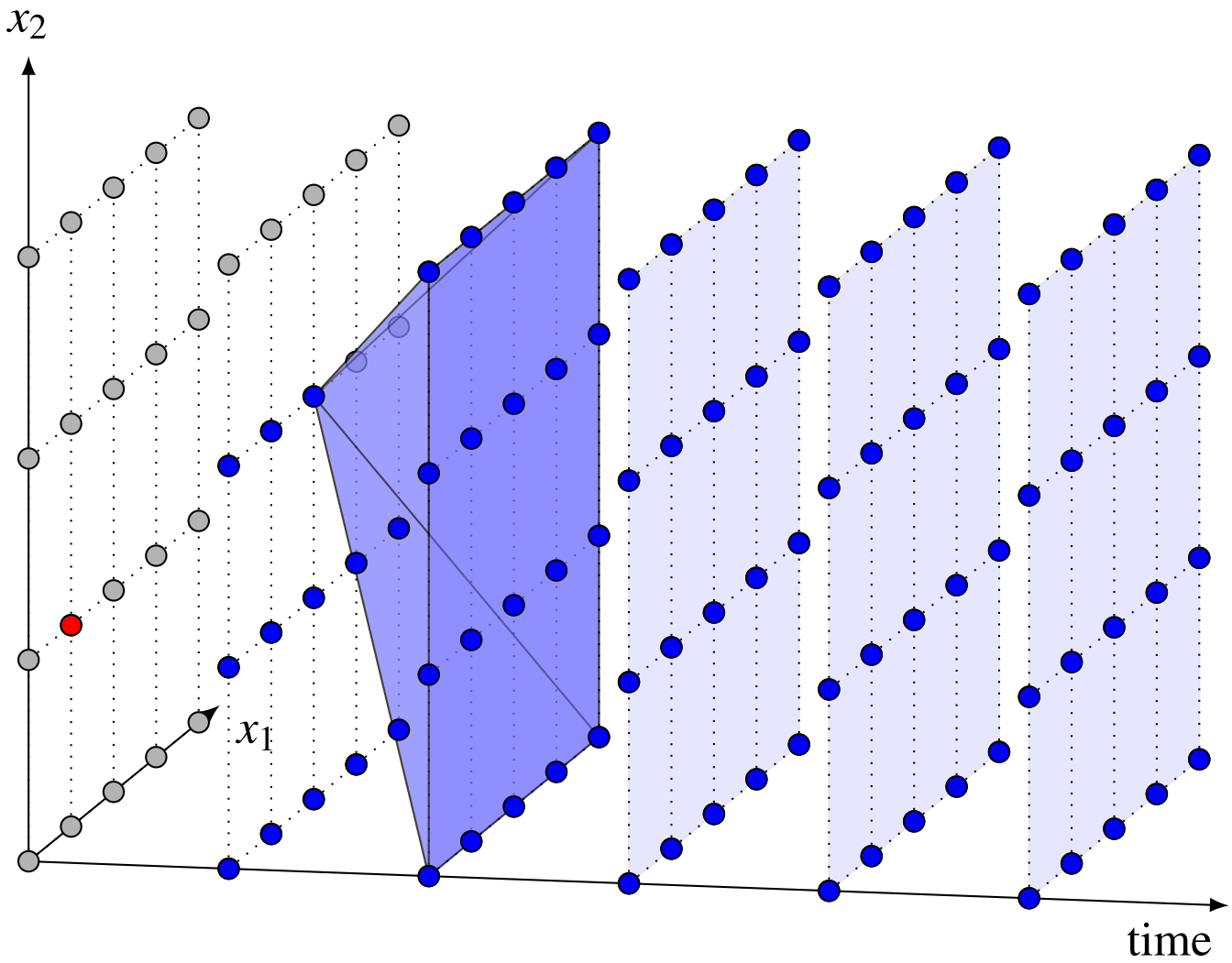
# Dynamic Programming: Illustration



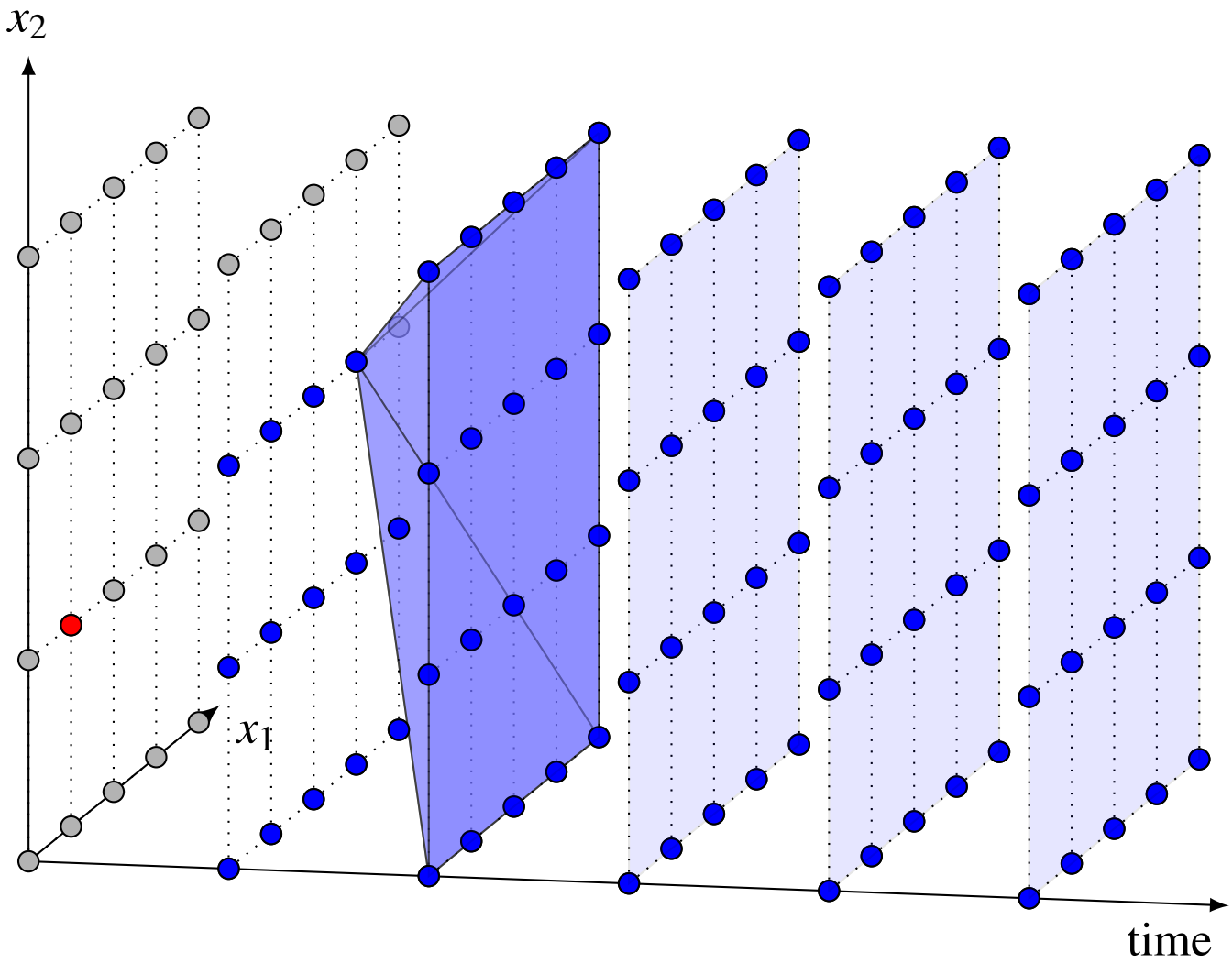
# Dynamic Programming: Illustration



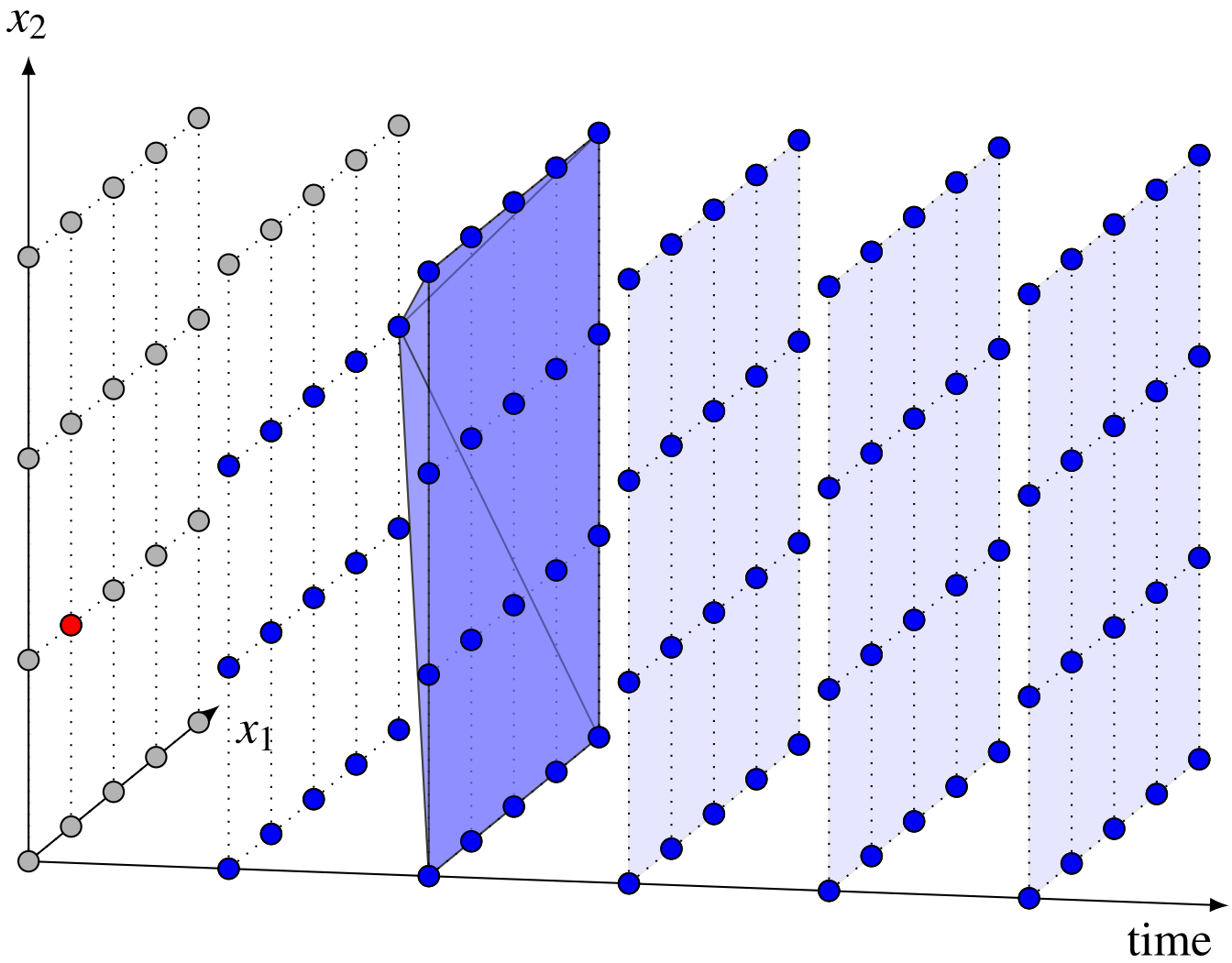
# Dynamic Programming: Illustration



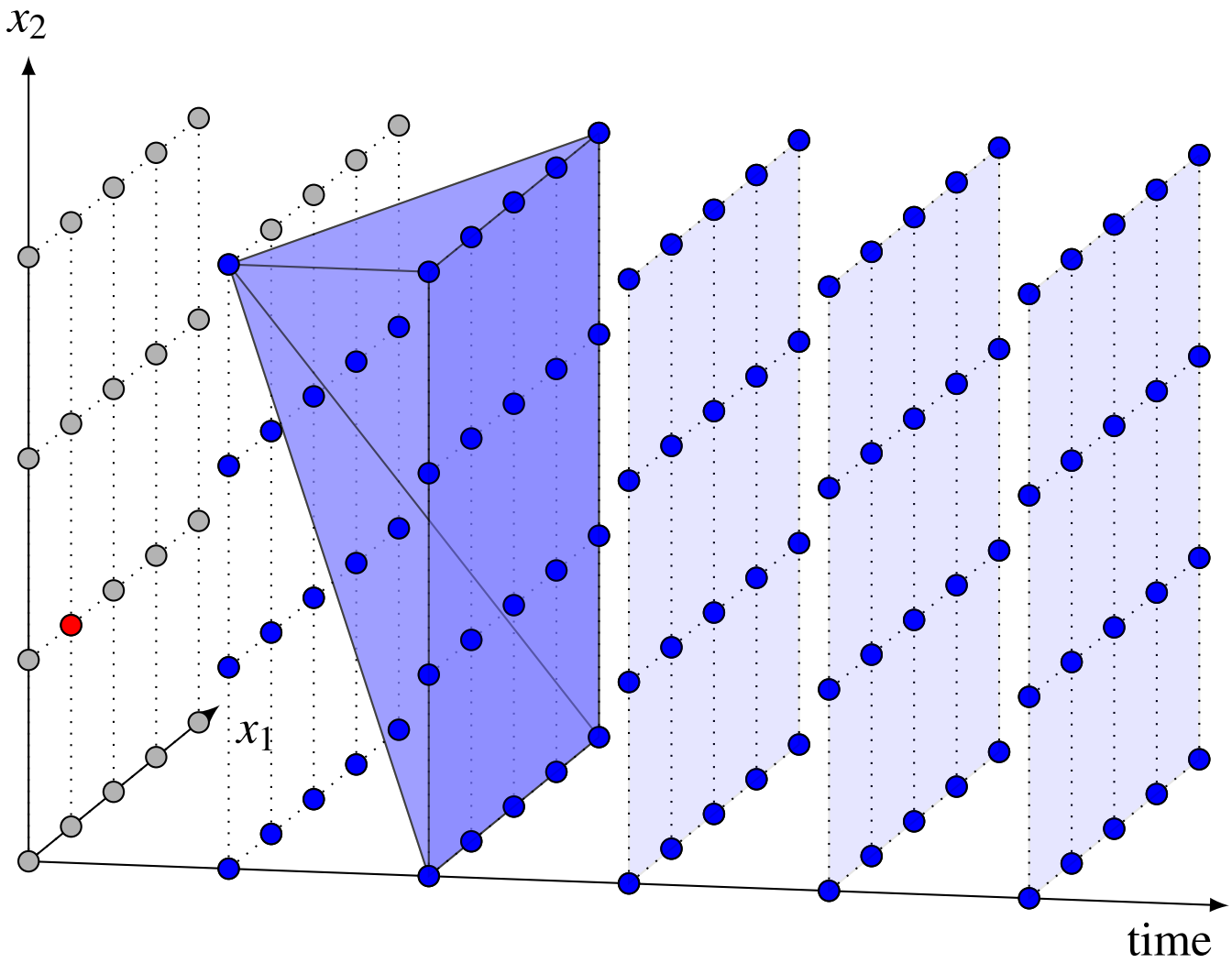
# Dynamic Programming: Illustration



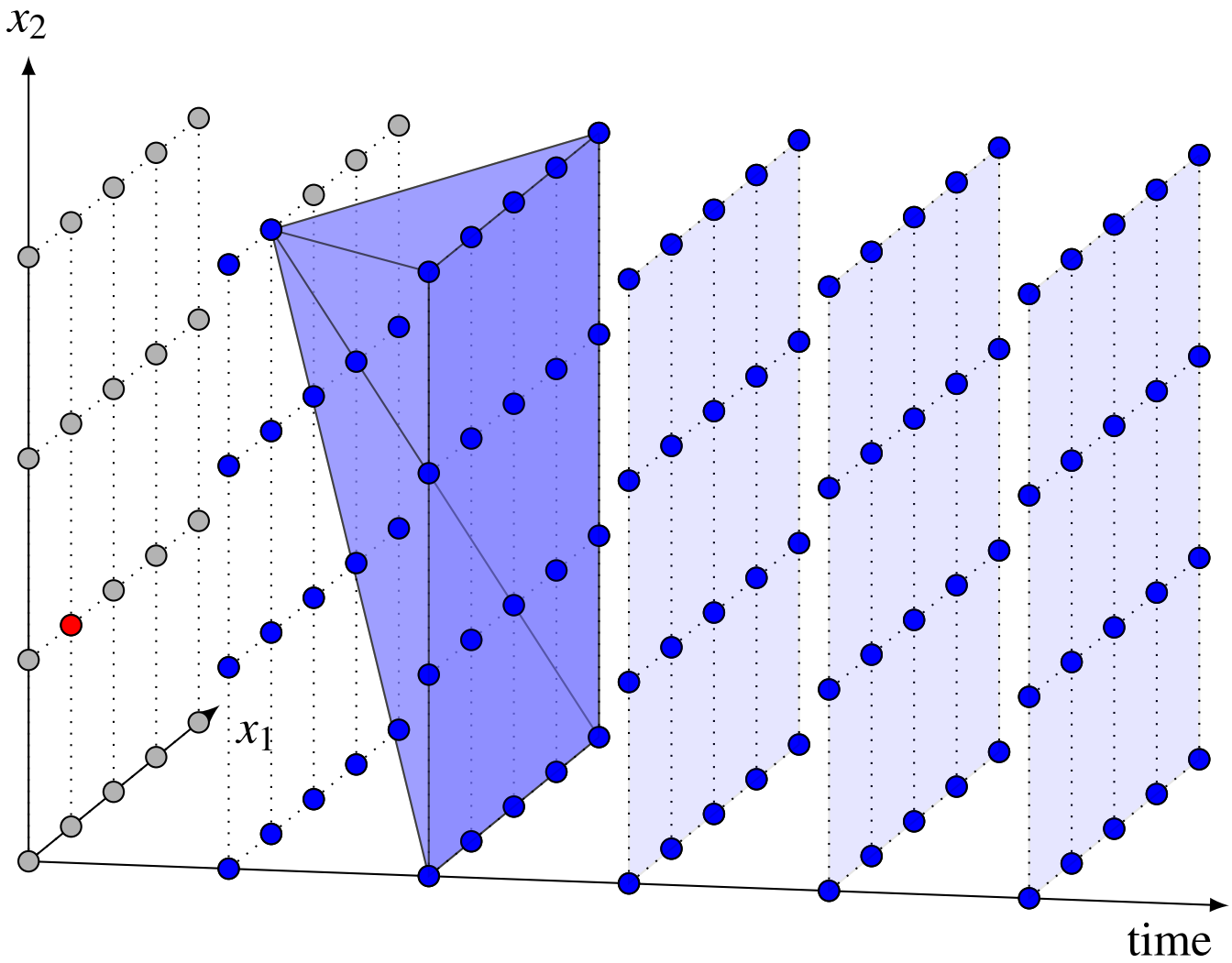
# Dynamic Programming: Illustration



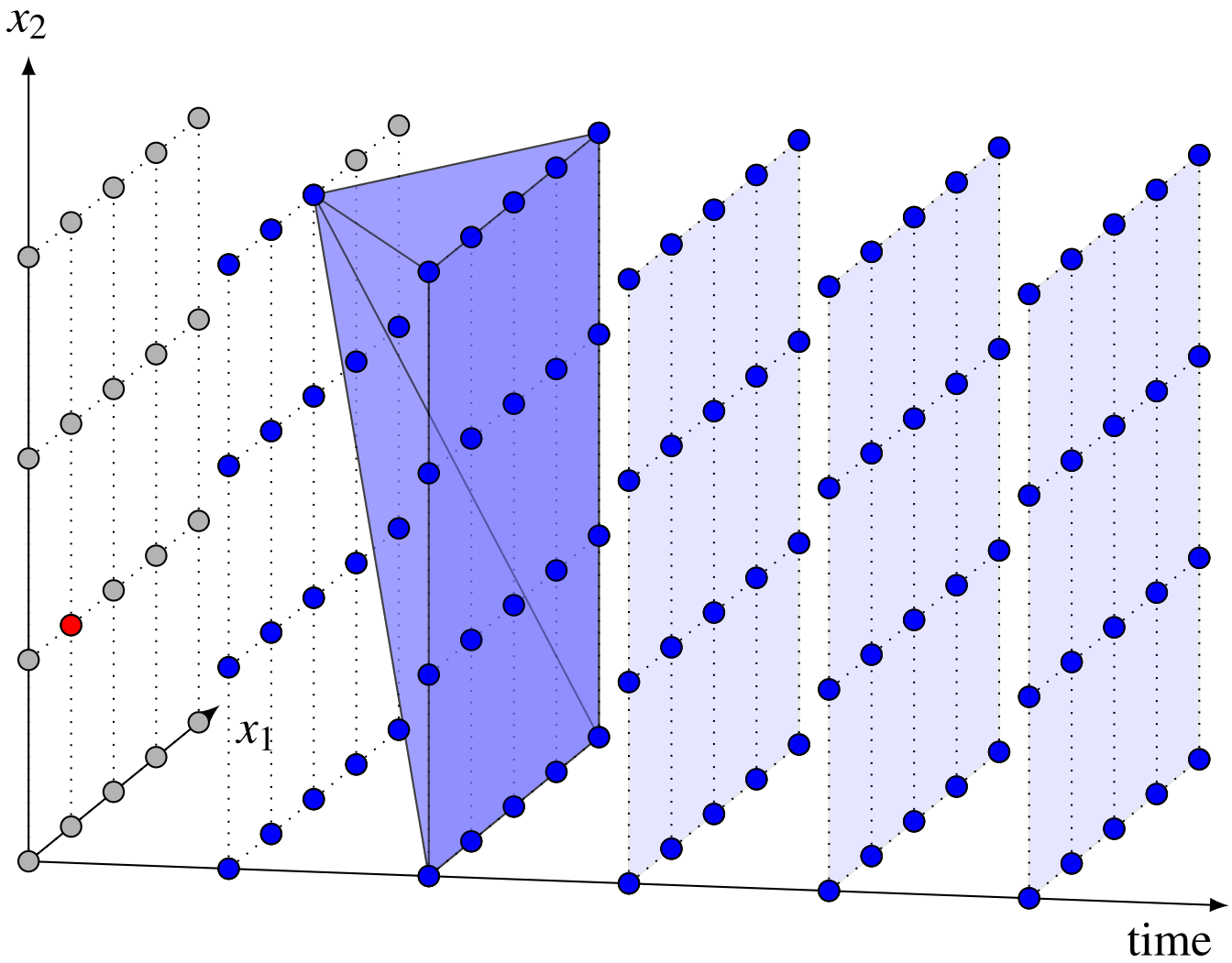
# Dynamic Programming: Illustration



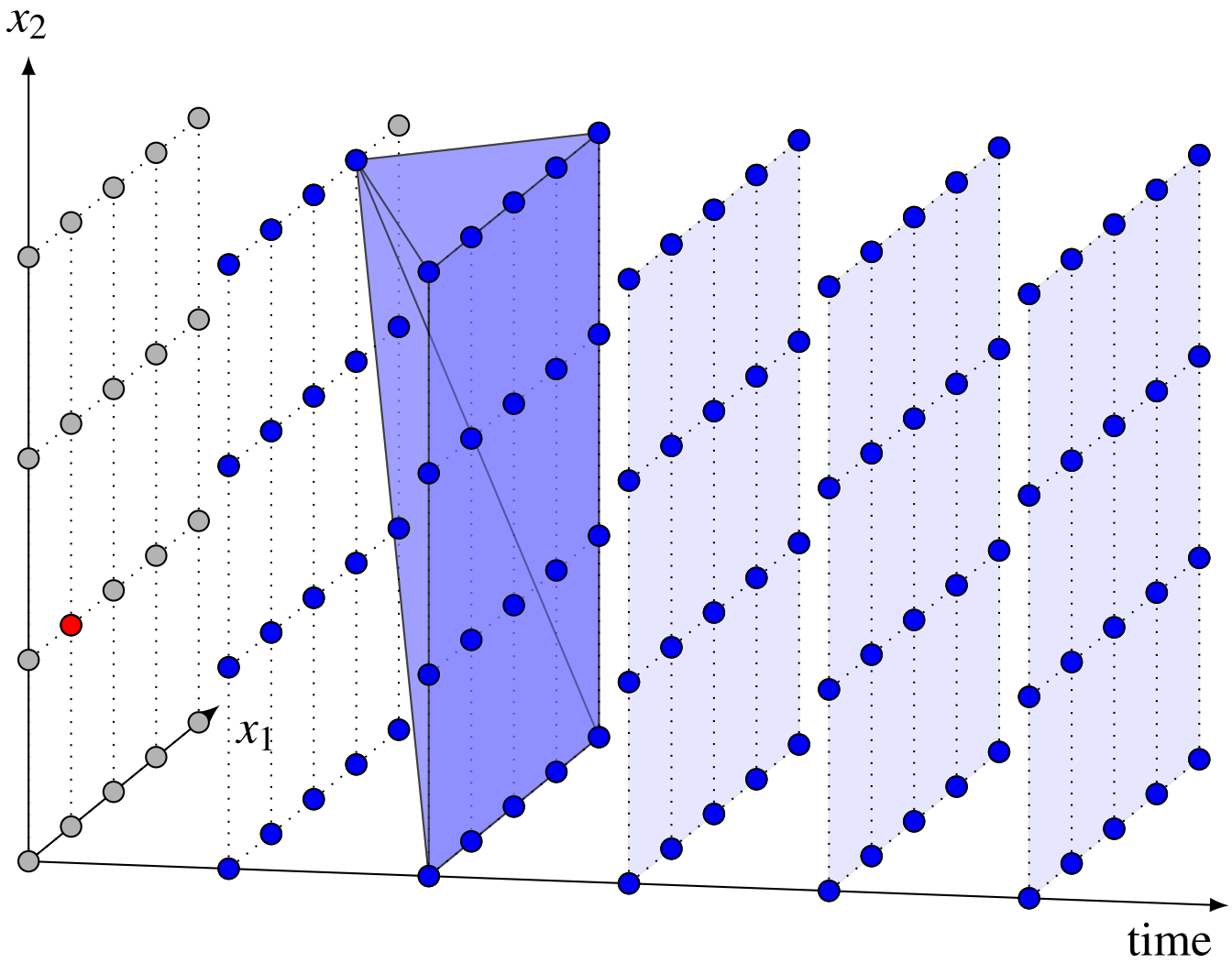
# Dynamic Programming: Illustration



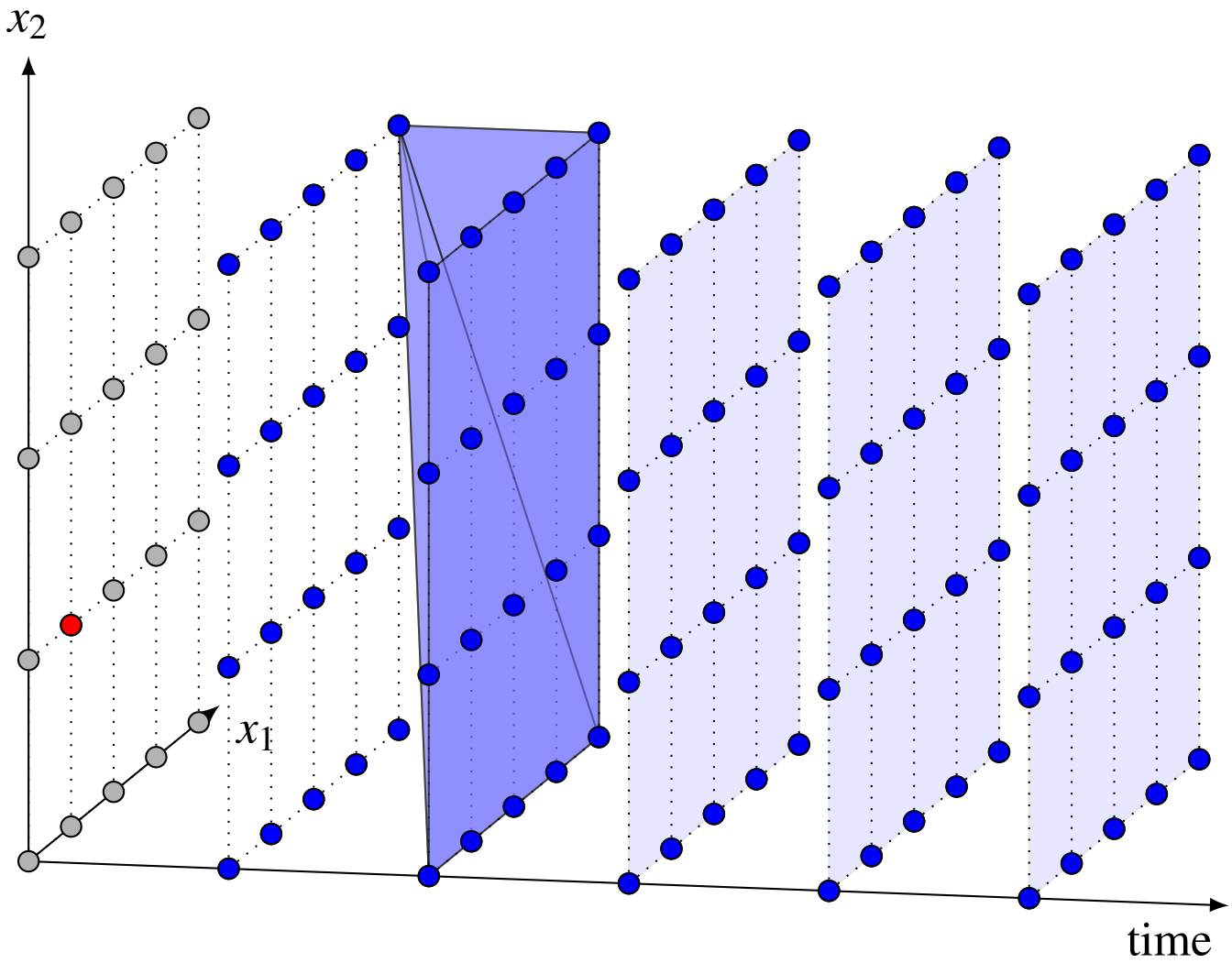
# Dynamic Programming: Illustration



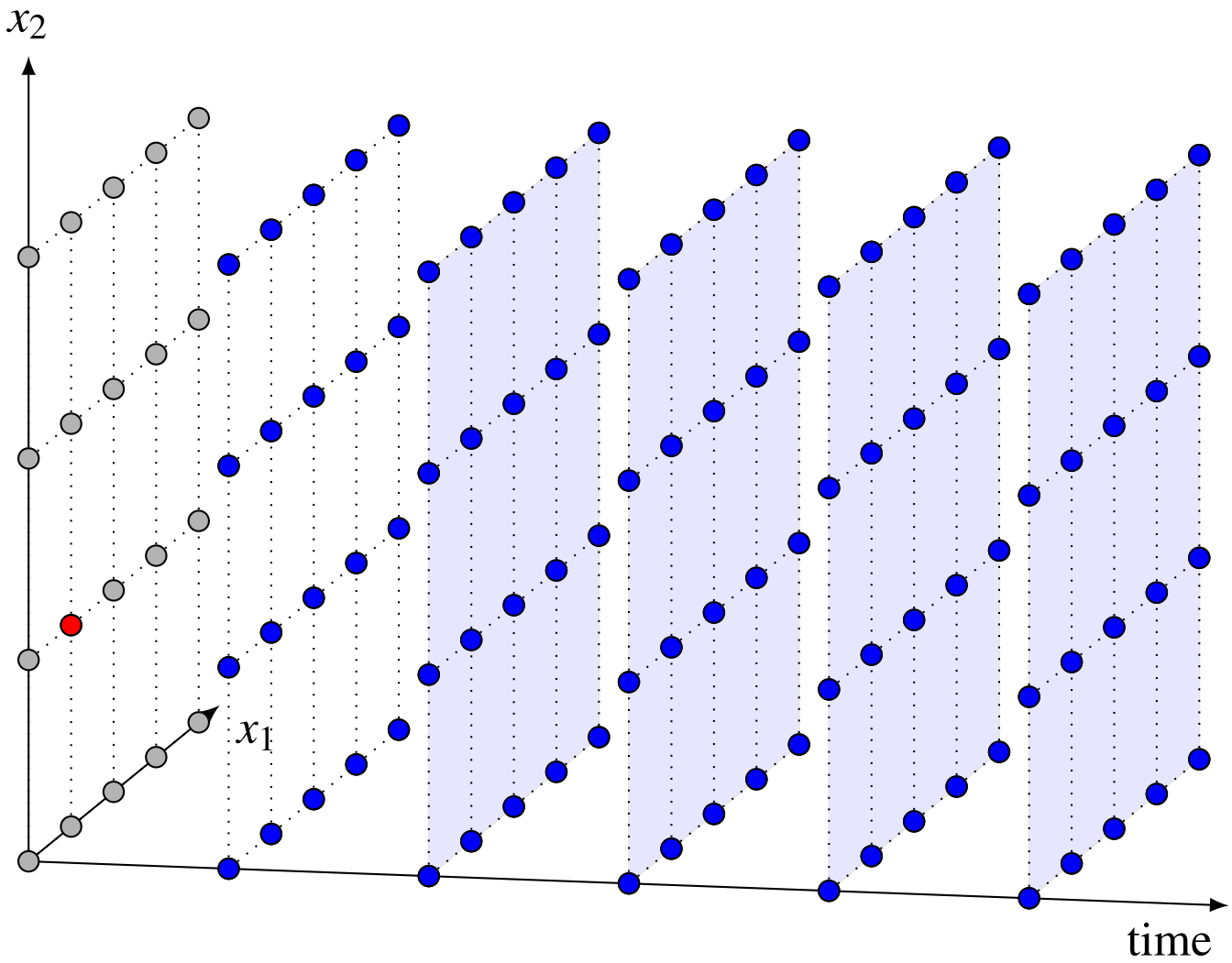
# Dynamic Programming: Illustration



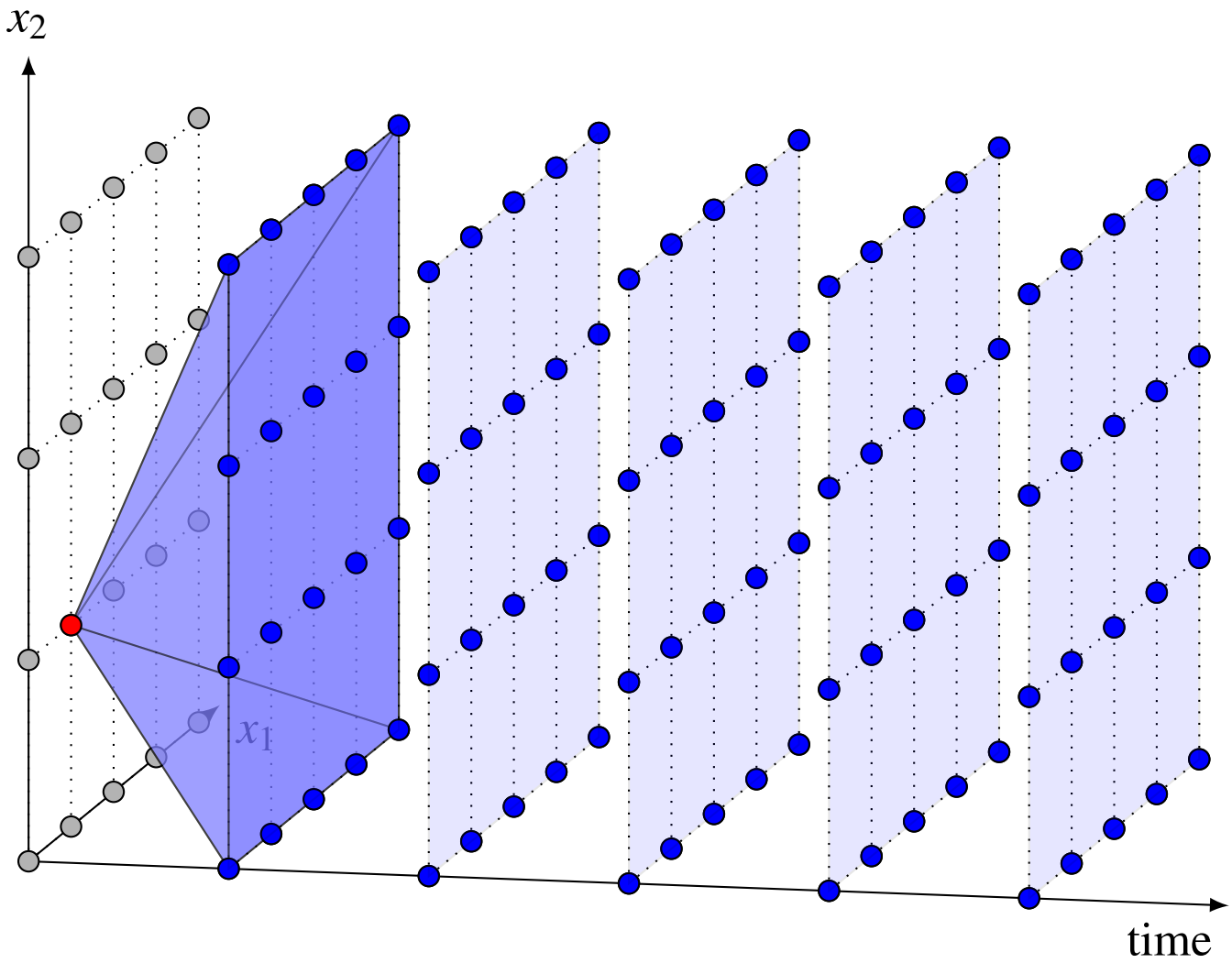
# Dynamic Programming: Illustration



# Dynamic Programming: Illustration



# Dynamic Programming: Illustration



# 3 curses of dimensionality

Complexity $= O(T \times |\mathbb{X}_t| \times |\mathbb{U}_t| \times |\Xi_t|)$ is linear in the number of time steps, but we have 3 curses of dimensionality :

1. State. Complexity is exponential in the dimension of $\mathbb{X}_t$
   e.g. 3 independent states each taking 10 values leads to a loop over 1000 points.
2. Decision. Complexity is exponential in the dimension of $\mathbb{U}_t$.
   $\rightsquigarrow$ due to exhaustive minimization of inner problem.
   Can be accelerated using faster method (e.g. MILP solver).
3. Alea. Complexity is exponential in the dimension of $\Xi_t$.
   $\rightsquigarrow$ due to expectation computation.
   Can be accelerated through Monte-Carlo approximation (still at least 1000 points)

In practice DP is not used for state of dimension more than 5.

# 3 curses of dimensionality

Complexity $= O(T \times |\mathbb{X}_t| \times |\mathbb{U}_t| \times |\Xi_t|)$ is linear in the number of time steps, but we have 3 curses of dimensionality :

1. State. Complexity is exponential in the dimension of $\mathbb{X}_t$
   e.g. 3 independent states each taking 10 values leads to a loop over 1000 points.
2. Decision. Complexity is exponential in the dimension of $\mathbb{U}_t$.
   $\rightsquigarrow$ due to exhaustive minimization of inner problem.
   Can be accelerated using faster method (e.g. MILP solver).
3. Alea. Complexity is exponential in the dimension of $\Xi_t$.
   $\rightsquigarrow$ due to expectation computation.
   Can be accelerated through Monte-Carlo approximation (still at least 1000 points)

In practice DP is not used for state of dimension more than 5.

# Illustrating the curse of dimensionality

We are in dimension 5 (not so high in the world of big data!) with 52 timesteps (common in energy management) plus 5 controls and 5 independent noises.

1. We discretize each state's dimension in 100 values: $|\mathbb{X}_t| = 100^5 = 10^{10}$
2. We discretize each control's dimension in 100 values: $|\mathbb{U}_t| = 100^5 = 10^{10}$
3. We use optimal quantization to discretize the noises' space in 10 values: $|\Xi_t| = 10$

Number of flops: $\mathcal{O}(52 \times 10^{10} \times 10^{10} \times 10) \approx \mathcal{O}(10^{23})$.
In the TOP500, the best computer computes $10^{17}$ flops/s.
Even with the most powerful computer, it takes at least 12 days to solve this problem.

# Numerical considerations

- The DP equation holds in (almost) any case.
- The algorithm shown before compute a look-up table of control for every possible state offline. It is impossible to do if the state is (partly) continuous.
- Alternatively, we can focus on computing offline an approximation of the value function $V_t$ and derive the optimal control online by solving a one-step problem, solved only at the current state :

$$\pi_t(x) \in \underset{u \in U_t(x)}{\arg\min}\, \mathbb{E}\Big[L_t(x, u, \xi_{t+1}) + V_{t+1} \circ f_t(x, u, \xi_{t+1})\Big]$$

- The field of Approximate DP gives methods for computing those approximate value function.
- The simpler one consisting in discretizing the state, and then interpolating the value function.

# Numerical considerations

- The DP equation holds in (almost) any case.
- The algorithm shown before compute a look-up table of control for every possible state offline. It is impossible to do if the state is (partly) continuous.
- Alternatively, we can focus on computing offline an approximation of the value function $V_t$ and derive the optimal control online by solving a one-step problem, solved only at the current state :

$$\pi_t(x) \in \underset{u \in U_t(x)}{\arg\min}\, \mathbb{E}\Big[L_t(x, u, \xi_{t+1}) + V_{t+1} \circ f_t(x, u, \xi_{t+1})\Big]$$

- The field of Approximate DP gives methods for computing those approximate value function.
- The simpler one consisting in discretizing the state, and then interpolating the value function.

# Numerical considerations

- The DP equation holds in (almost) any case.
- The algorithm shown before compute a look-up table of control for every possible state offline. It is impossible to do if the state is (partly) continuous.
- Alternatively, we can focus on computing offline an approximation of the value function $V_t$ and derive the optimal control online by solving a one-step problem, solved only at the current state :

$$\pi_t(x) \in \underset{u \in U_t(x)}{\arg\min}\, \mathbb{E}\Big[L_t(x, u, \boldsymbol{\xi}_{t+1}) + V_{t+1} \circ f_t(x, u, \boldsymbol{\xi}_{t+1})\Big]$$

- The field of Approximate DP gives methods for computing those approximate value function.
- The simpler one consisting in discretizing the state, and then interpolating the value function.

# Dynamic Programming : Discretization-Interpolation

**Algorithm 2:** Dynamic Programming Algorithm (Continuous)

**Data:** Problem parameters, discretization,
one-stage solver, interpolation operator;
**Result:** approximation of optimal value;

$\tilde{V}_T \equiv K$ ;
**for** $t : T-1 \to 0$ **do**
  **for** $x \in \mathbb{X}_t^D$ **do**
    $\tilde{V}_t(x) := \min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{\xi}_{t+1}) + \tilde{V}_{t+1} \circ f_t(x, u, \boldsymbol{\xi}_{t+1})\Big]$;
  Define $\tilde{V}_t$ by interpolating $\{\tilde{V}_t(x) \mid x \in \mathbb{X}_t^D\}$;

The strategy obtained is given by

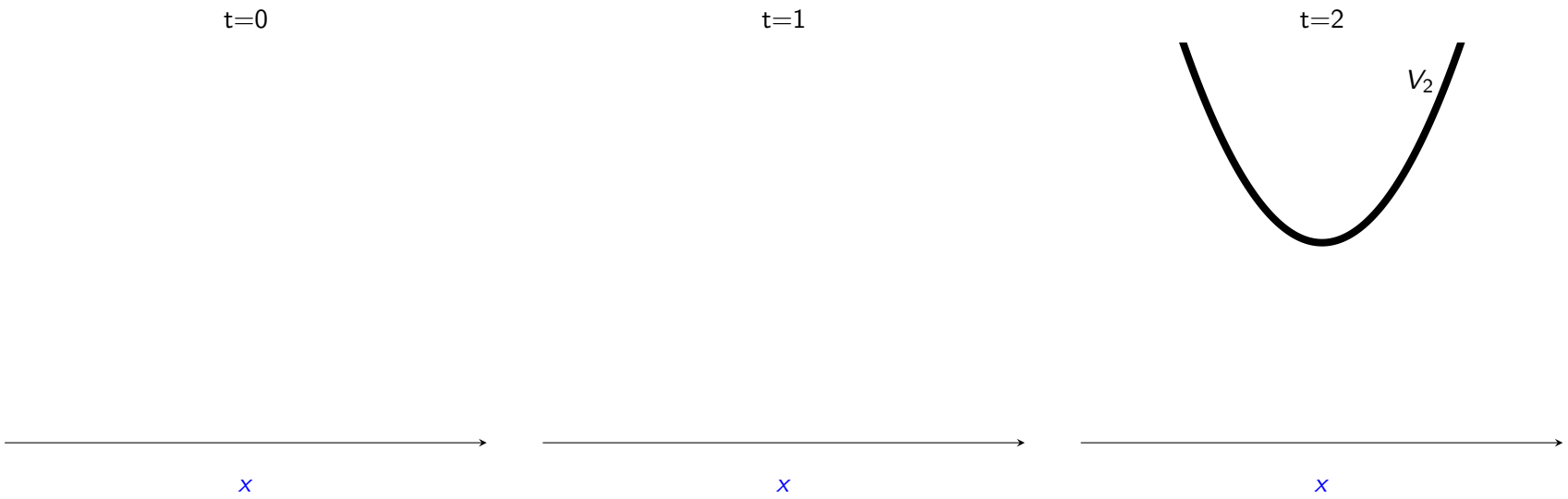
$$\pi_t(x) \in \arg\min_{u \in U_t(x)} \mathbb{E}\Big[L_t(x, u, \boldsymbol{\xi}_{t+1}) + \tilde{V}_{t+1} \circ f_t(x, u, \boldsymbol{\xi}_{t+1})\Big].$$

# Presentation Outline

# Dynamic Programming : continuous and convex case

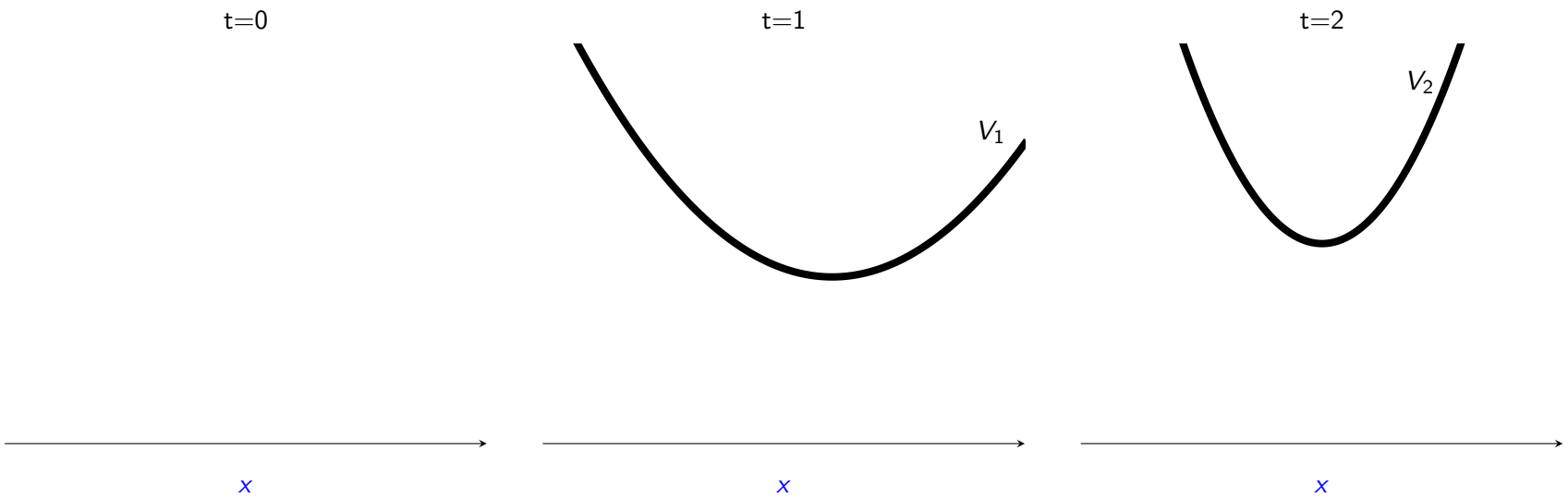
- If the problem has continuous states and control the classical approach consists in discretizing.
- With further assumption on the problem (convexity, linearity) we can look at a dual approach:
  - Instead of discretizing and interpolating the Bellman function we choose to do a polyhedral approximation.
  - Indeed we choose a "smart state" in which we compute the value of the function and its marginal value (tangeant).
  - Knowing that the problem is convex and using the power of linear solver we can efficiently approximate the Bellman function.
- This approach is known as SDDP in the electricity community and widely used in practice.

# SDDP



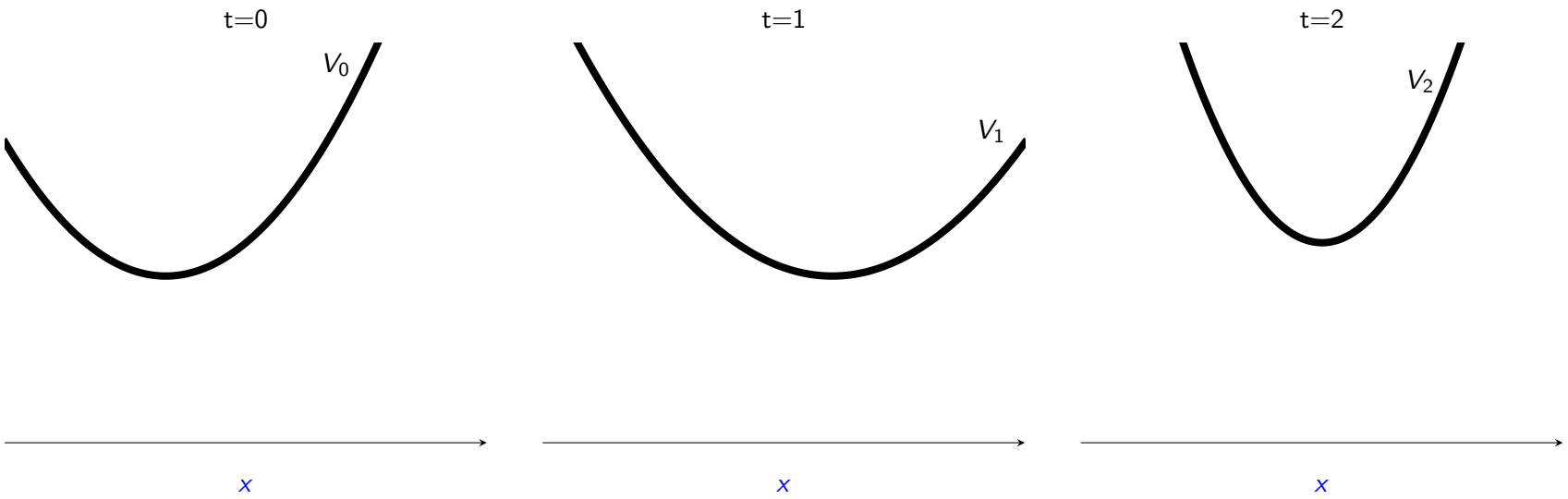

Final Cost $V_2 = V_2$

# SDDP



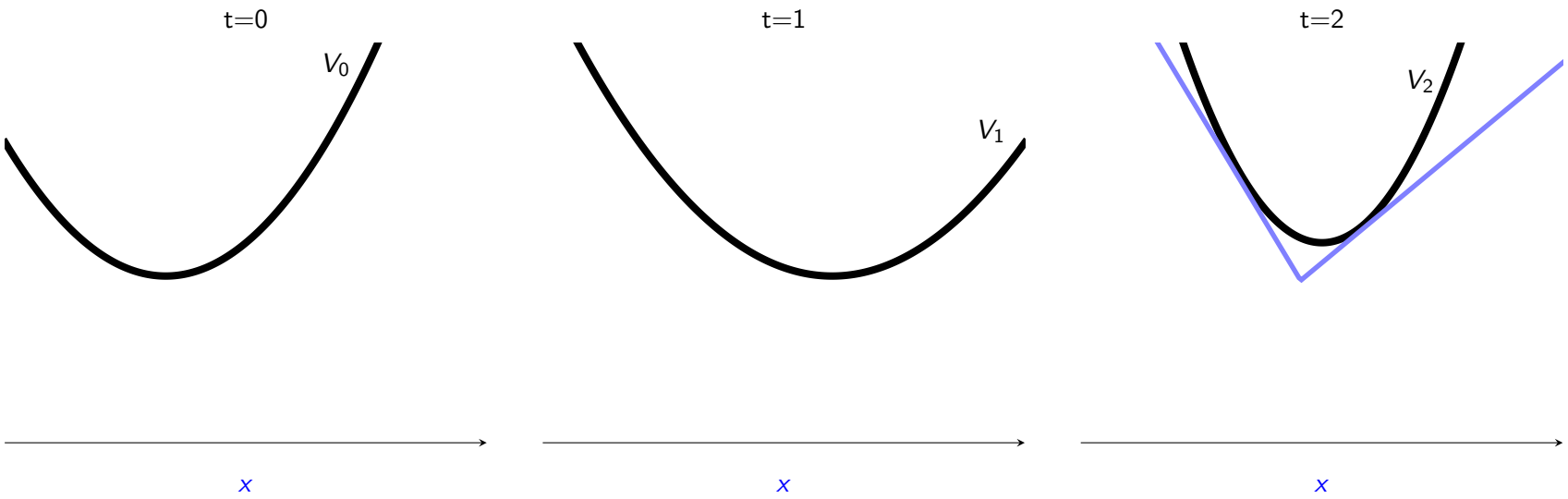

Real Bellman function $V_1 = \mathcal{B}_1(V_2)$

# SDDP



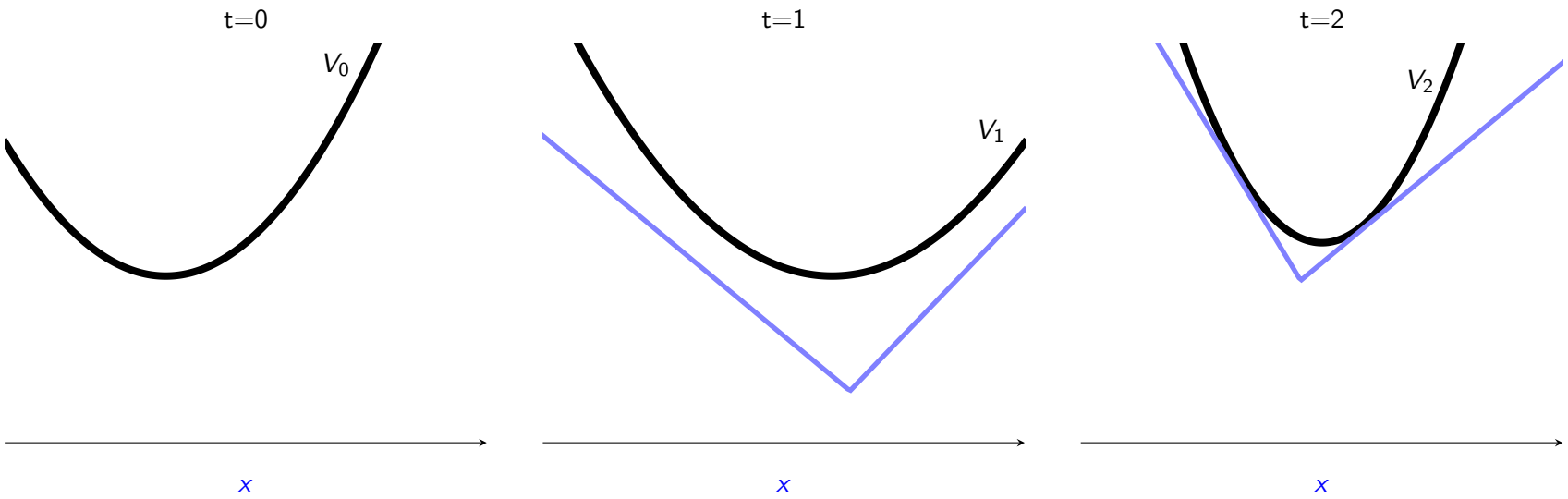

Real Bellman function $V_0 = \mathcal{B}_0(V_1)$

# SDDP



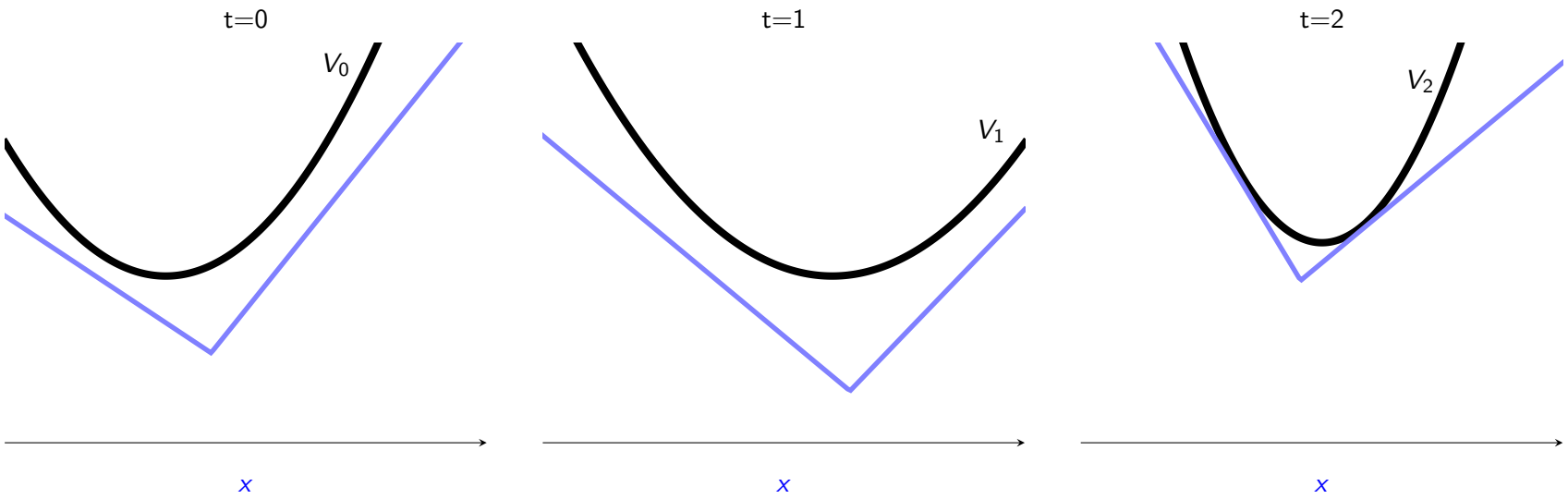

Lower polyhedral approximation $\underline{V_2}$ of $V_2$

# SDDP



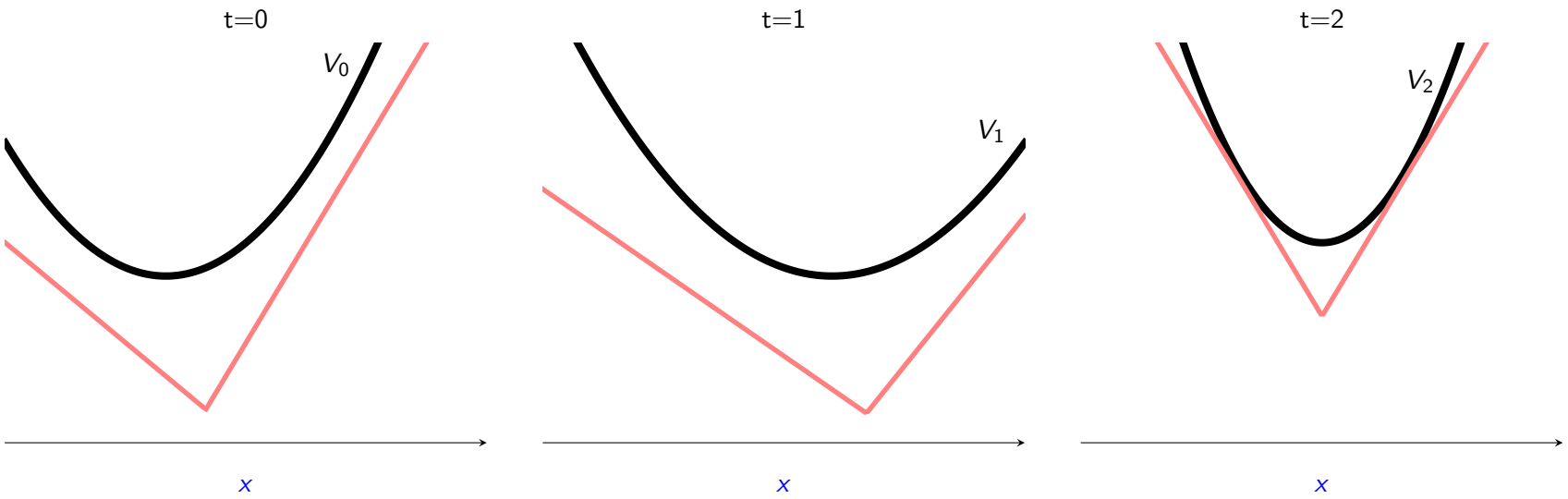

Lower polyhedral approximation $\underline{V}_1 = \mathcal{B}_t(\underline{V}_2)$ of $V_1$

# SDDP



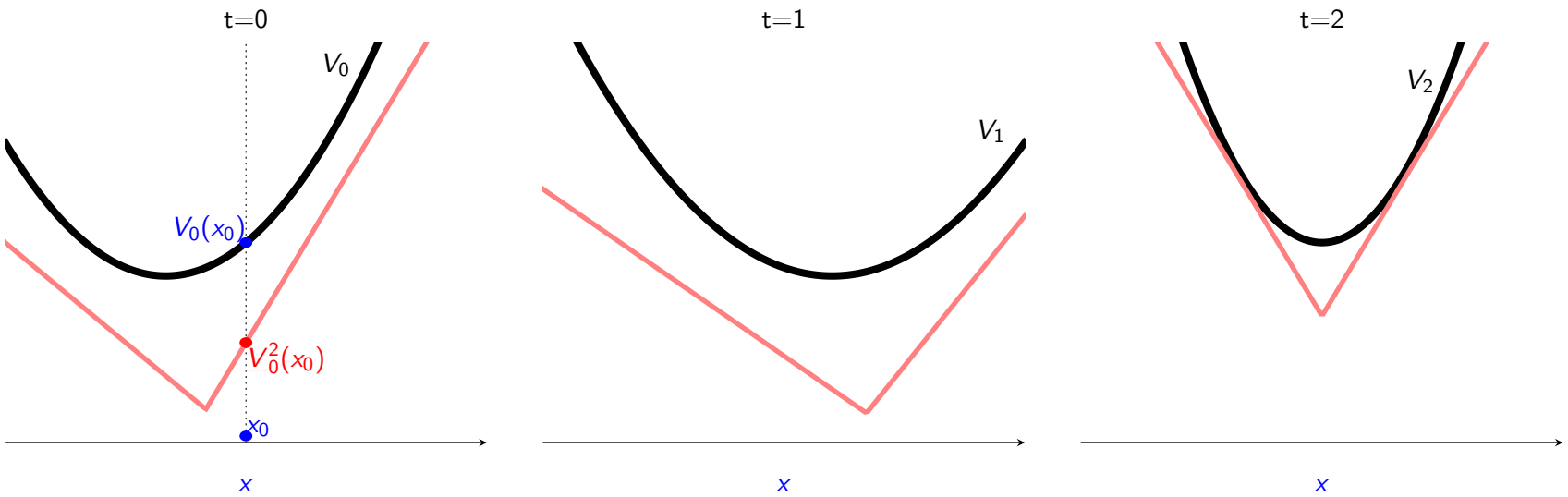

Lower polyhedral approximation $\underline{V}_0 = \mathcal{B}_t(\underline{V}_1)$ of $V_0$

# SDDP



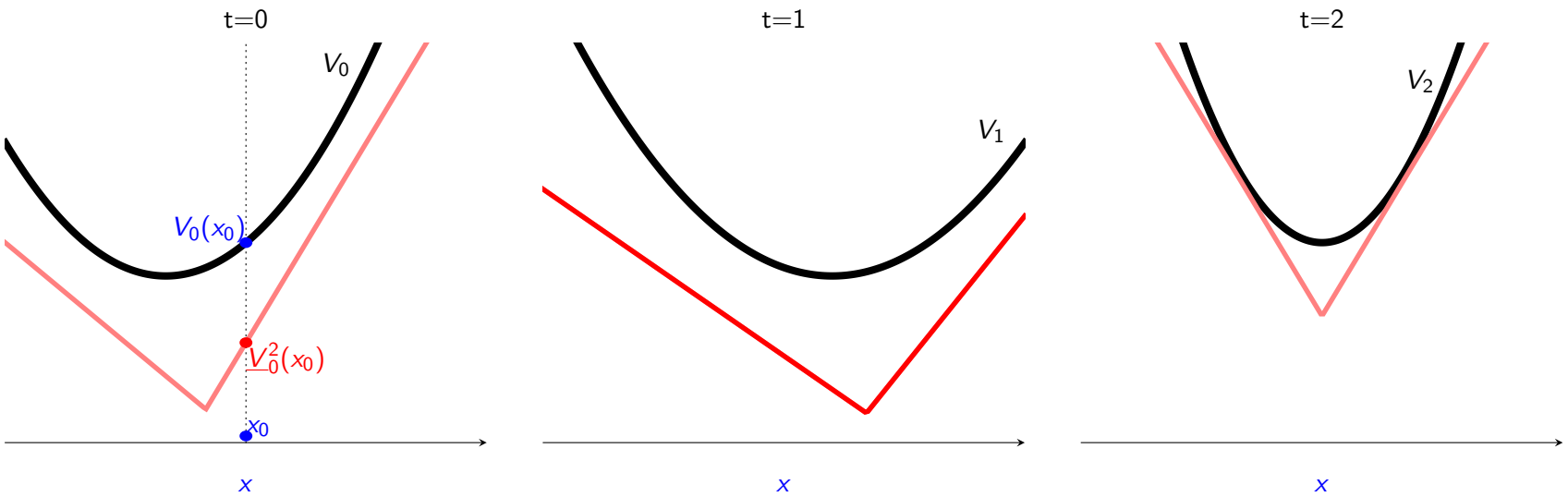

Assume that we have lower polyhedral approximations of $V_t$

# SDDP



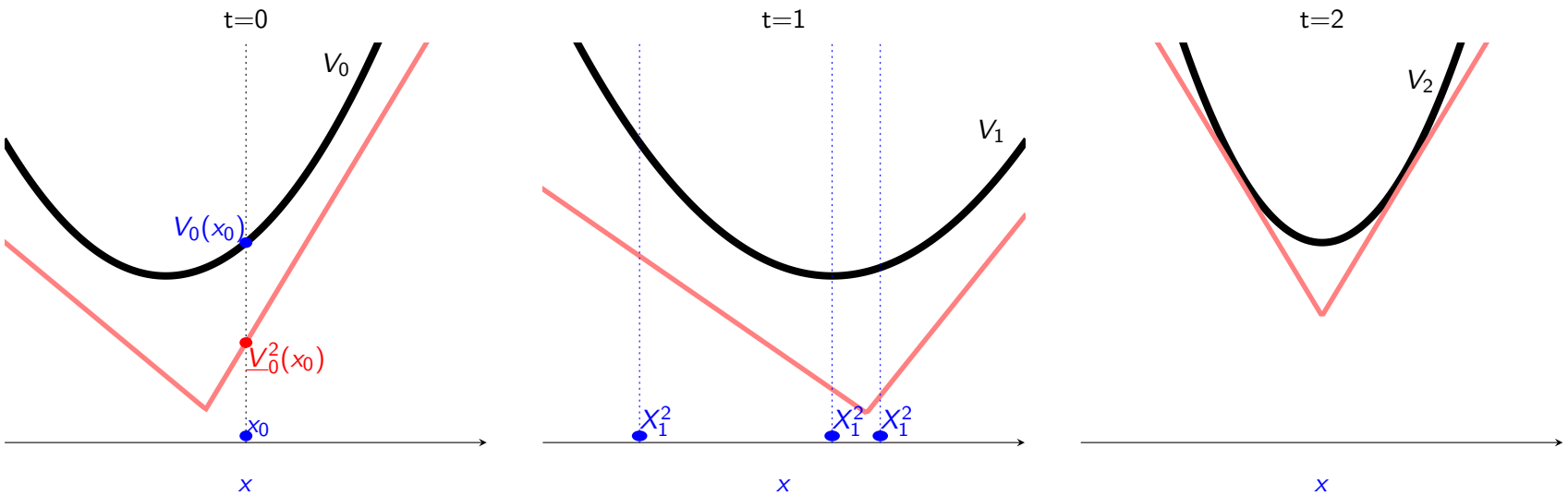

Obtain a lower bound on the value of our problem

# SDDP



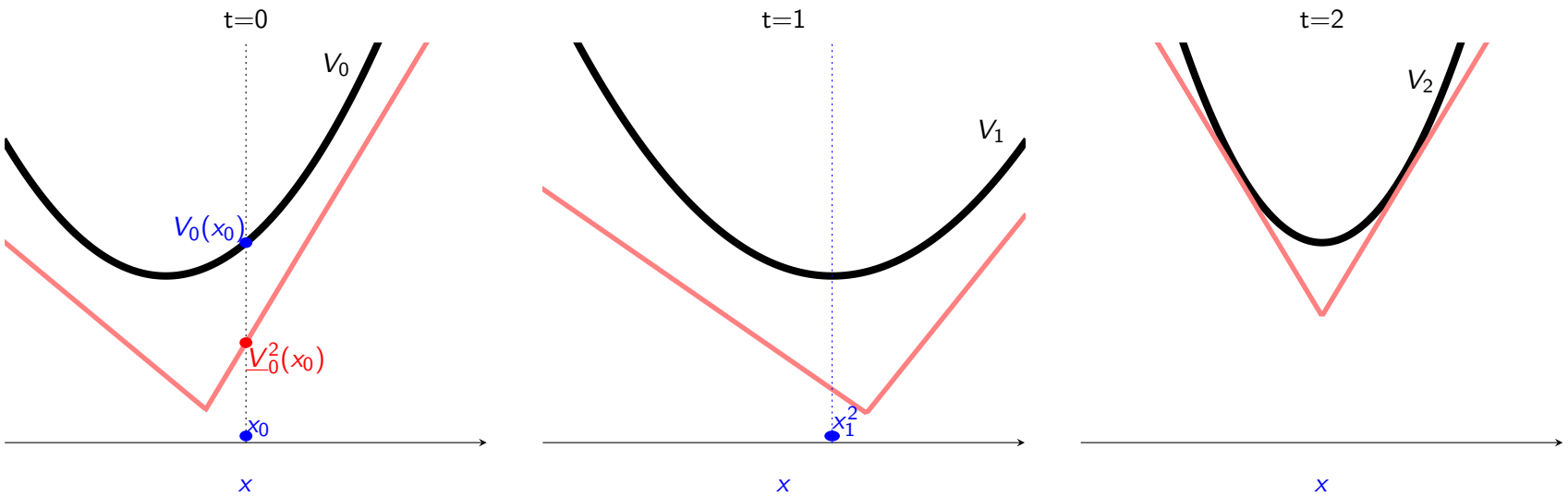

Apply $\mathcal{F}_0(\underline{V}_1^{(2)})(x_0)$ and obtain $\boldsymbol{X}_1^{(2)}$

# SDDP



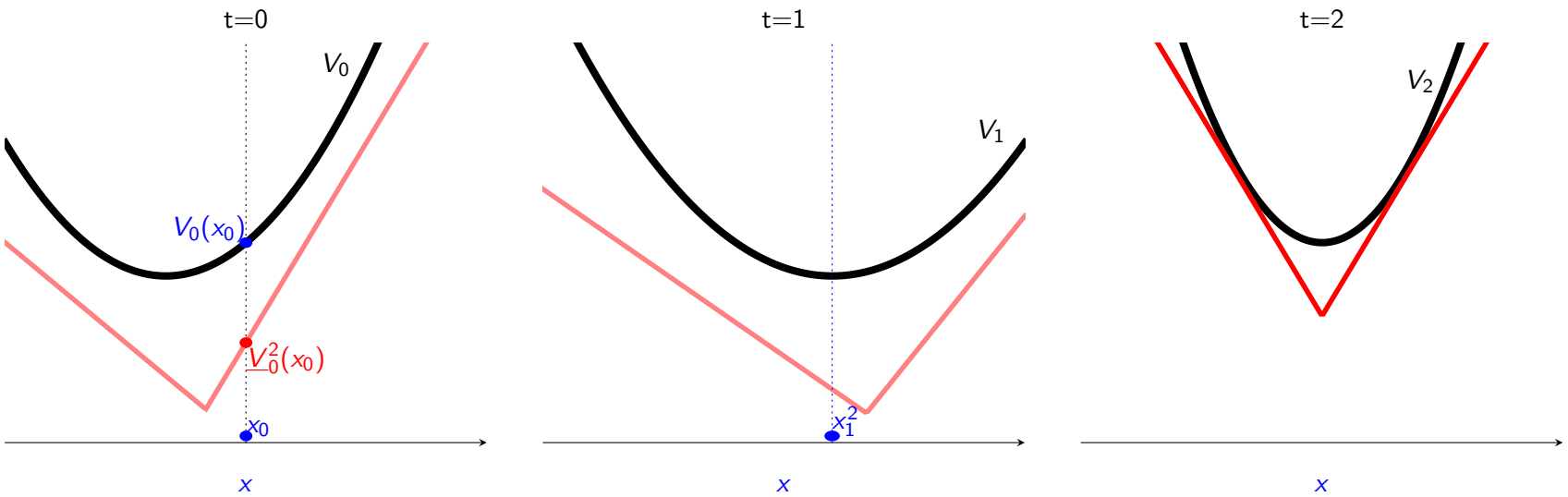

Apply $\mathcal{F}_0(\underline{V}_1^{(2)})(x_0)$ and obtain $\boldsymbol{X}_1^{(2)}$

# SDDP



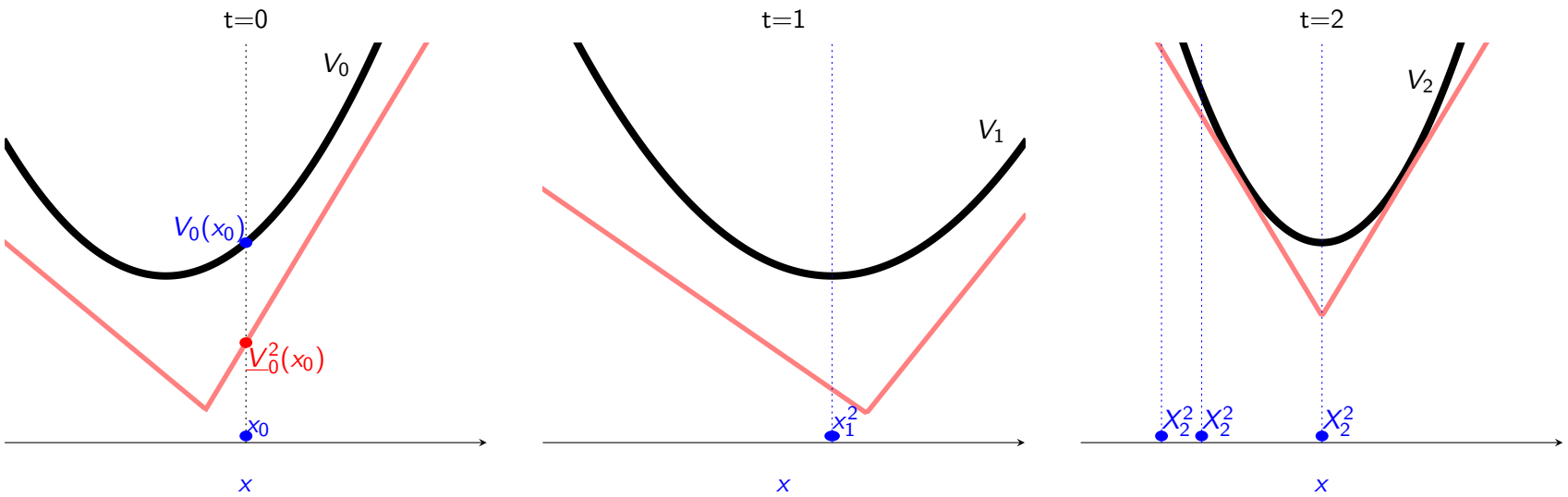

Draw a random realisation $x_1^{(2)}$ of $\boldsymbol{X}_1^{(2)}$

# SDDP



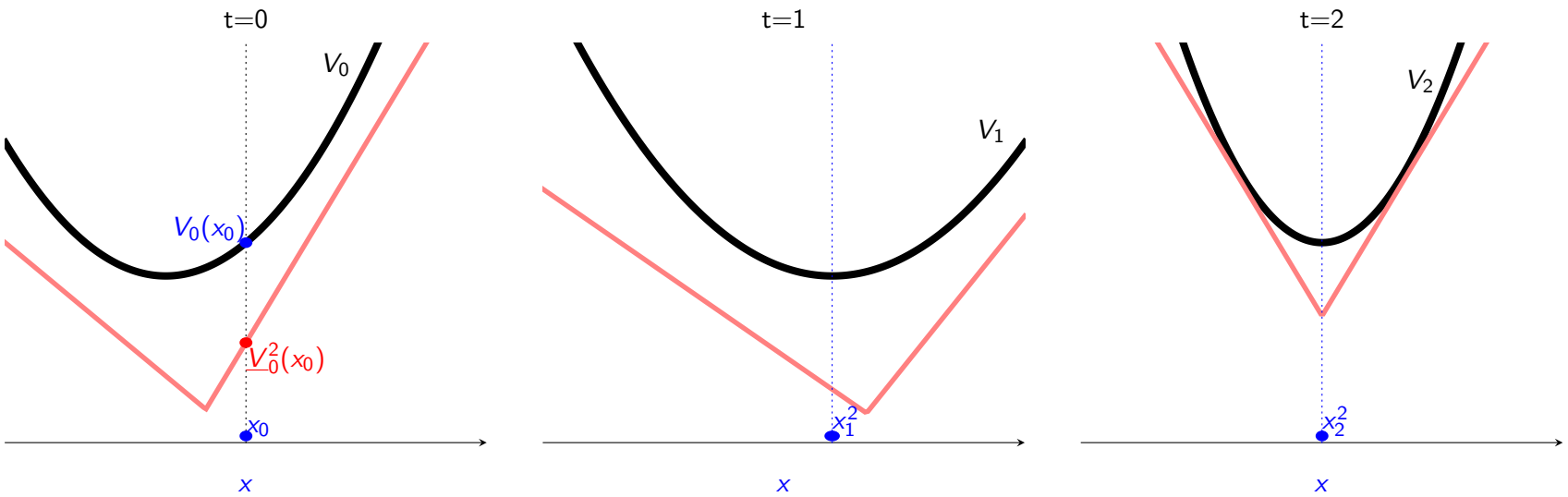

We apply $\mathcal{F}_1(\underline{V}_1^{(2)})(x_1^{(2)})$ and obtain $\boldsymbol{X}_2^{(2)}$

# SDDP



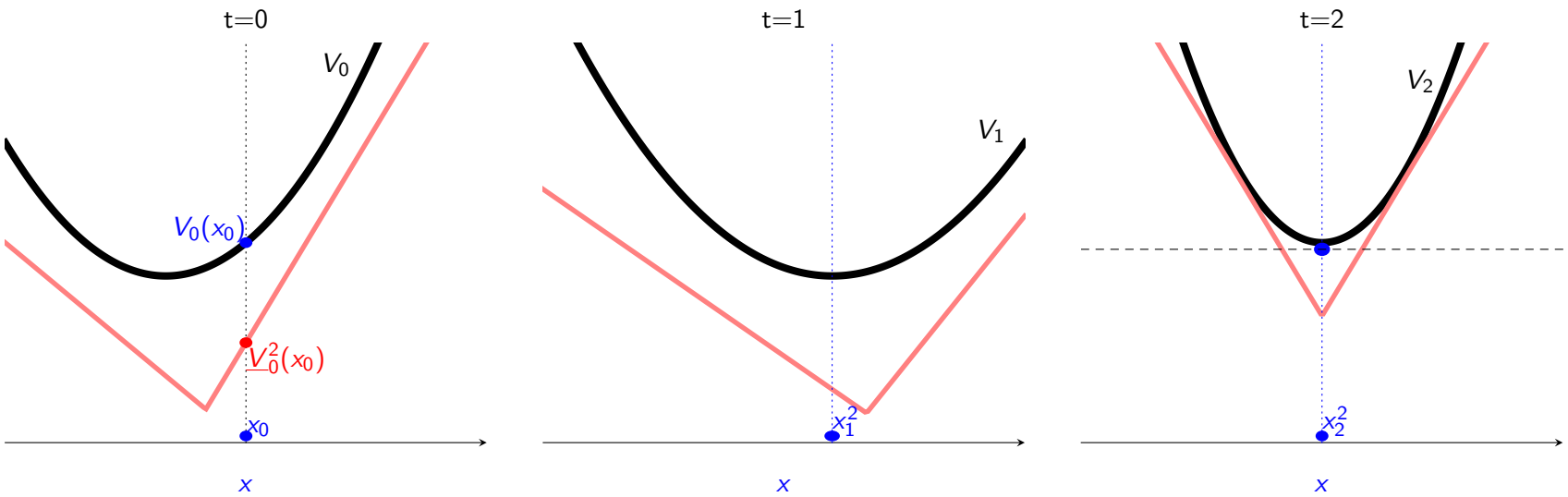

We apply $\mathcal{F}_1(\underline{V}_1^{(2)})(x_1^{(2)})$ and obtain $\boldsymbol{X}_2^{(2)}$

# SDDP



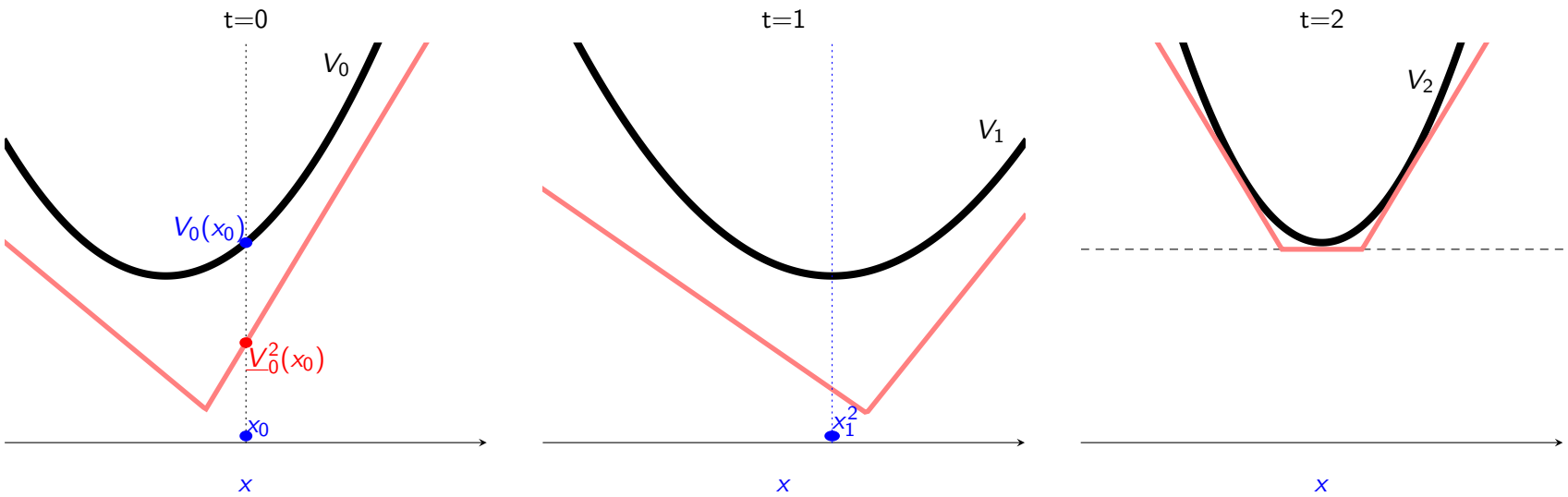

Draw a random realisation $x_2^{(2)}$ of $\boldsymbol{X}_2^{(2)}$

# SDDP



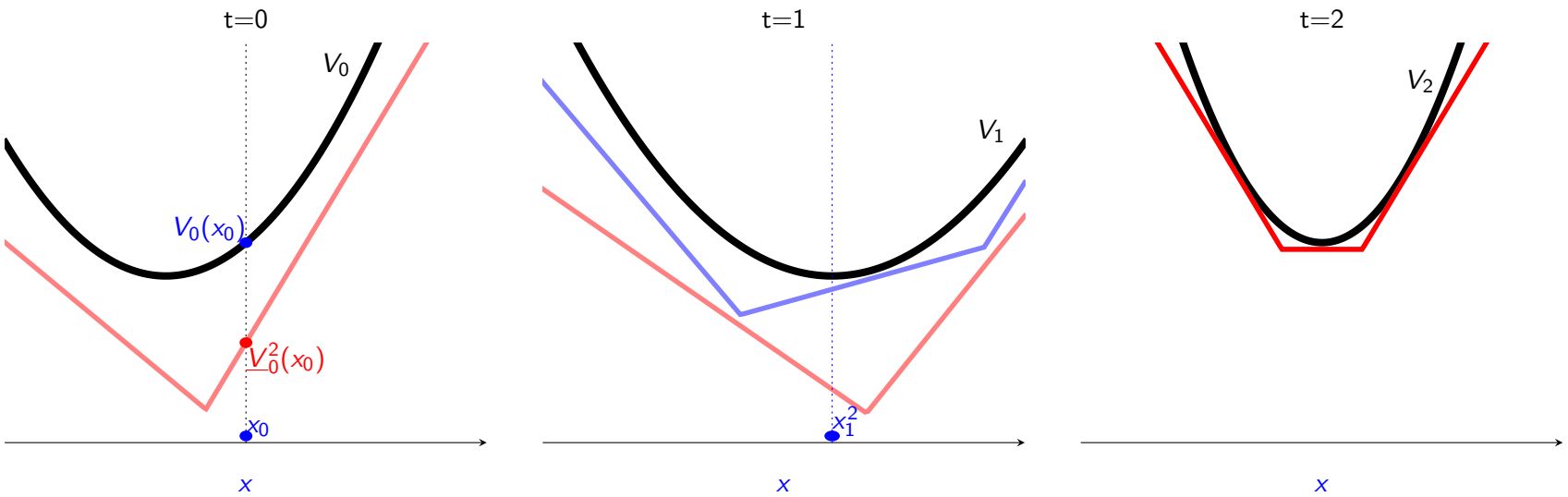

Compute a cut for $V_2$ at $x_2^{(2)}$

# SDDP



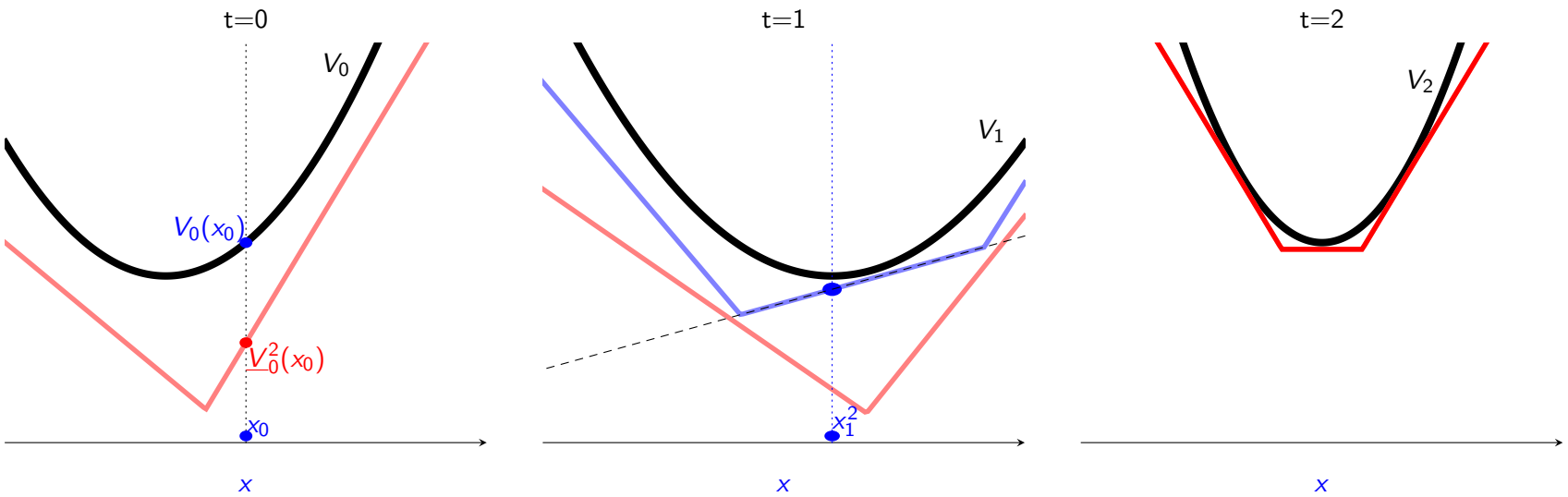

Add the cut to $\underline{V}_2^{(2)}$ which gives $\underline{V}_2^{(3)}$

# SDDP



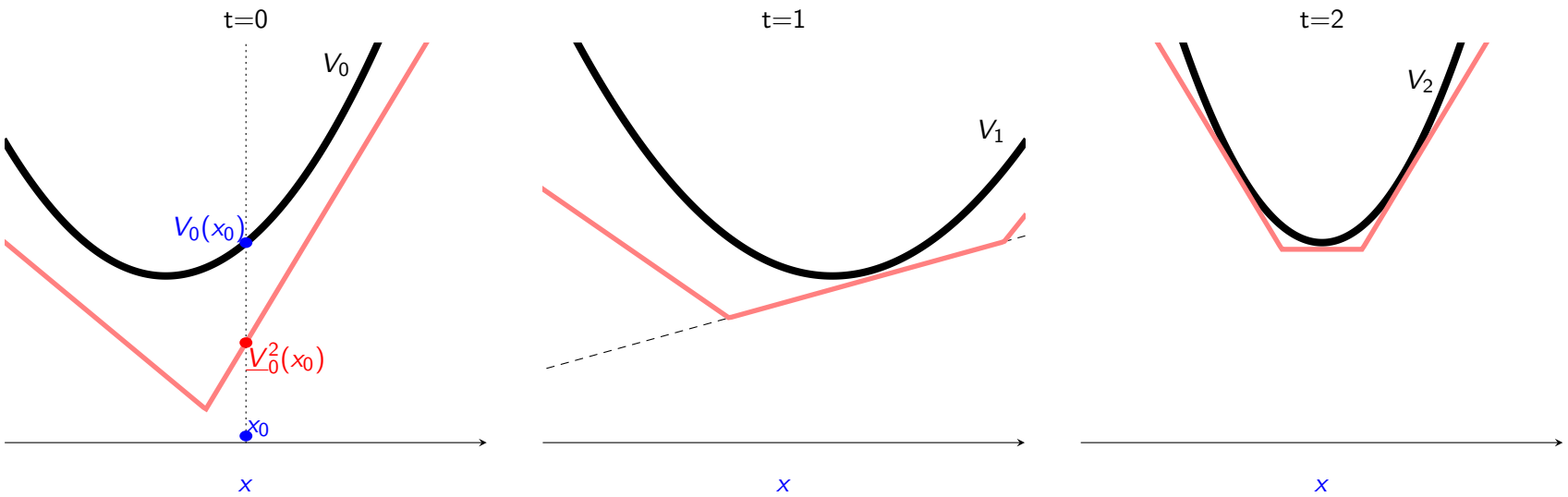

A new lower approximation of $V_1$ is $\mathcal{B}_1(\underline{V}_2^{(3)})$

# SDDP



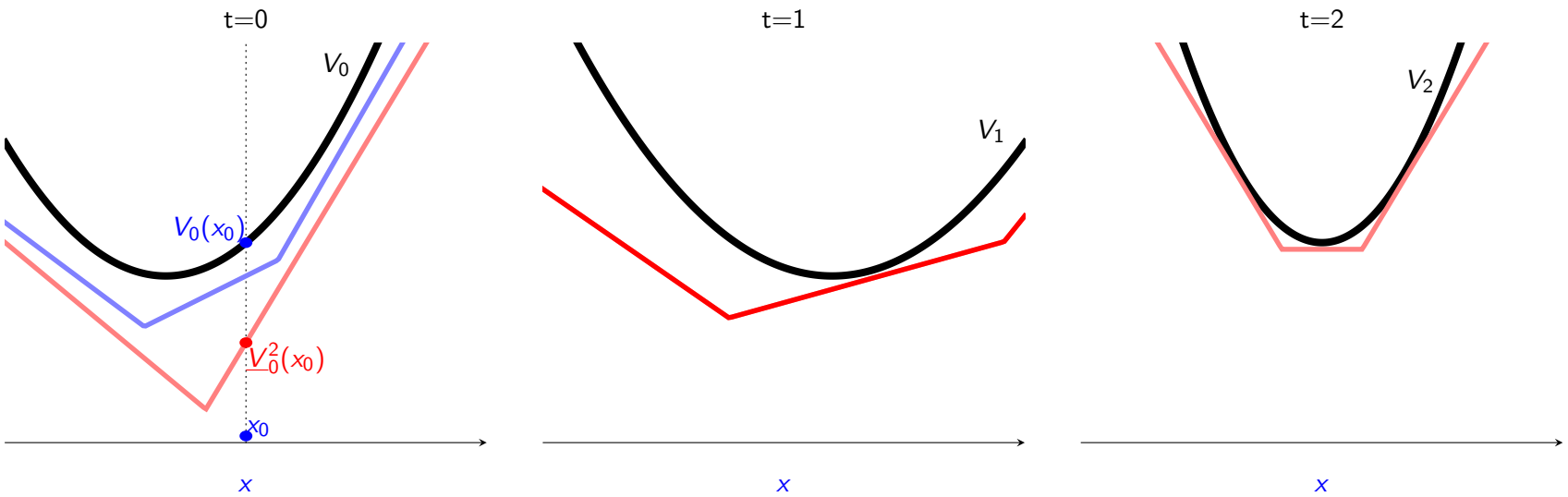

Compute the face active at $x_1^{(2)}$

# SDDP



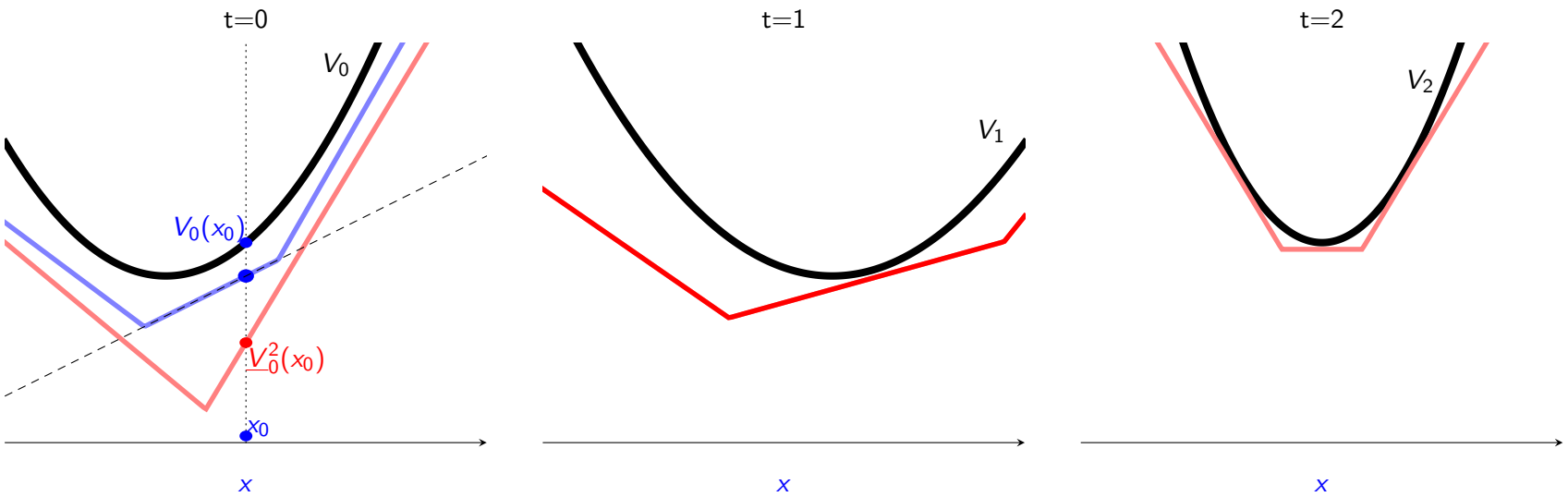

Add the cut to $\underline{V}_1^{(2)}$ which gives $\underline{V}_1^{(3)}$

# SDDP



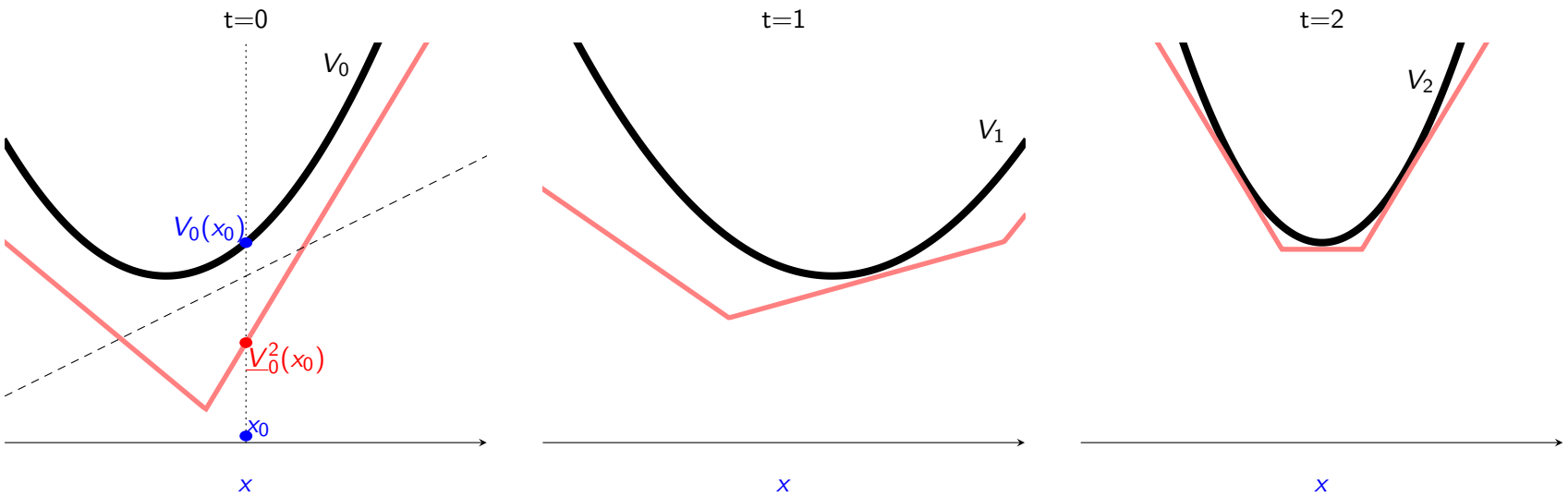

A new lower approximation of $V_0$ is $\mathcal{B}_0(\underline{V}_1^{(3)})$

# SDDP



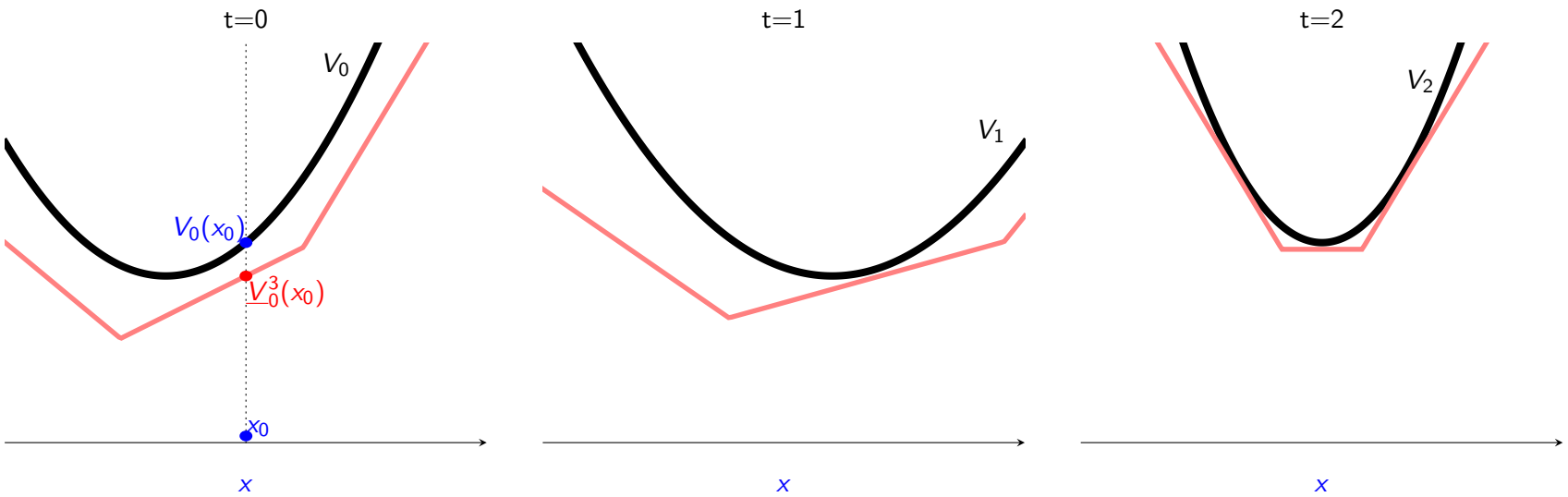

Compute the face active at $x_0$

# SDDP




Compute the face active at $x_0$

# SDDP




Obtain a new lower bound

# Presentation Outline

# Numerical Limits of SP and SDP

**Stochastic Programming:**

- $\mathcal{O}(n_u n_\xi^T)$ variables.
- In practice 2 or 3 information steps are the limit.
- Hence we need to approximate the information structure.
- Often rely on linearity of the costs and dynamic function.
- Mainly return an estimation of the optimal cost and the first step control.

**Dynamic Programming:**

- Requires Markovian assumption.
- Number of elementary operation is $\mathcal{O}(T|\mathbb{X}_t|^{n_X}|\mathbb{U}_t|^{n_U}|\Xi_t|^{n_\xi})$.
- Limited to dimension 4 or 5.
- Can use convexity and linearity assumption to increase dimension.
- Return value function, that can be used to derive optimal policy.

# Numerical Limits Illustration

- Dam example in SP approach
  - 5 interconnected dams
  - 5 controls per timesteps
  - 52 timesteps (one per week, over one year)
  - $n_\xi = 10$ noises for each timestep

  $\rightsquigarrow 10^{52}$ scenarios, and $\approx 5.10^{52}$ constraints in the extensive formulation. ($10^{24}$ bytes on internet).
- Dam example in DP approach
  - Each state discretized in 100 values: $|\mathbb{X}_t|^{n_X} = 100^5 = 10^{10}$
  - Each control discretized in 100 values: $|\mathbb{U}_t|^{n_U} = 100^5 = 10^{10}$
  - We use optimal quantization to discretize the noises' space in 10 values: $|\Xi_t| = 10$

  Number of flops: $\mathcal{O}(52 \times 10^{10} \times 10^{10} \times 10) \approx \mathcal{O}(10^{23})$.

  $\rightsquigarrow$ around 12 days on best TOP500 computer.

# Presentation Outline

# What if my problem is... I

An investment problem for a supply network : where to open new production centers while not knowing yet the demand.

- Two type of control : where to open and how to operate.
- I am mainly interested in the question of "where to open".
- State dimension is important (number of possible units), demand is correlated in time.

# What if my problem is... I

An investment problem for a supply network : where to open new production centers while not knowing yet the demand.

- Two type of control : where to open and how to operate.
- I am mainly interested in the question of "where to open".
- State dimension is important (number of possible units), demand is correlated in time.

$\Longrightarrow$ Stochastic Programming approach is natural here. First step decision : where to open, second step : operation decision. The modelization is optimistic as it assume perfect knowledge of demand for the operational problem (i.e. once investment is decided).

# What if my problem is... II

A weekly stock management problem over a year, with random demand and known production costs.

- 52 time-steps, with more or less independent noise.
- Each time-step yield new information
- natural state (the stock)

# What if my problem is... II

A weekly stock management problem over a year, with random demand and known production costs.

- 52 time-steps, with more or less independent noise.
- Each time-step yield new information
- natural state (the stock)

$\Longrightarrow$ Dynamic Programming approach is natural here.

# What if my problem is... III

A Unit Commitment Problem, where you have to decide at $t = 0$ which unit will be producing at which time during the next 24 hours, and then adjust to satisfy the actual demand.

- 48 time step with new correlated information at each step (demand, renewable energy, breakdown...)
- Decision at $t = 0$ are important, operational decision will be recomputed.
- Operational decision are time-correlated (ramping constraints).
- State dimension is high

# What if my problem is... III

A Unit Commitment Problem, where you have to decide at $t = 0$ which unit will be producing at which time during the next 24 hours, and then adjust to satisfy the actual demand.

- 48 time step with new correlated information at each step (demand, renewable energy, breakdown...)
- Decision at $t = 0$ are important, operational decision will be recomputed.
- Operational decision are time-correlated (ramping constraints).
- State dimension is high

$\Longrightarrow$ Dynamic Programming approach requires physical (smaller state) and statistical (independent noise) approximations, Stochastic Programming requires informational approximation (knowing a breakdown far in advance). Hard choice, SP might be more appropriate.

# What if my problem is... IV

Long term energy investment problem like setting up a new line / a new production unit, in a region with huge hydroelectric stock (e.g. Brazil).

- 120 time-step, each with new information
- Some decisions at time $t = 0$ affect the whole problem, they are the one that interest us.
- Linear dynamic and costs
- Noise is time-correlated.

# What if my problem is... IV

Long term energy investment problem like setting up a new line / a new production unit, in a region with huge hydroelectric stock (e.g. Brazil).

- 120 time-step, each with new information
- Some decisions at time $t = 0$ affect the whole problem, they are the one that interest us.
- Linear dynamic and costs
- Noise is time-correlated.

$\Longrightarrow$ For the operational part Dynamic Programming (SDDP) is really adapted if we model the noise with AR process. SP requires a huge informational approximation.

# Any Alternatives ?

If neither SP nor SDP is suited to your problem, here are a few pointer to heuristics that can help:

- Assume that the Bellman Value function is given as a linear combination of basis function and fit the coefficient (look toward Approximate Dynamic Programming field).
- Assume that you are anticipative, compute the first control, apply it (throwing all other controls), observe your current state and solve again (Open Loop Feedback Control).
- Assume that you have a two-stage problem, compute the first control, apply it (throwing all other controls), observe your current state and solve again (Repeated Two-Stage Programming approach).
- Decompose your problem (L-shaped method, Progressive Hedging, DADP...)
- ...

# Presentation Outline

# Conclusion

- Uncertain parameters requires careful manipulation
- Some questions has to be answered :
  - attitude toward risk
  - careful constraint formulation
  - information structure
- Numerically solving a stochastic problem require one of the two following assumption:
  - A small number of information steps : Stochastic Programming approach.
  - Time independence of noises : Dynamic Programming approach.
- Decomposition is tricky in stochastic setting, but can be very efficient.
- Don't forget to define a simulator to evaluate any solution you obtain from any approach.


Anymore questions for me?