Machine Learning Department
School of Computer Science
Carnegie Mellon University

MAP Inference with MILP

Matt Gormley
Lecture 12
Oct. 7, 2019

Reminders

- Homework 2: BP for Syntax Trees
  - Out: Sat, Sep. 28
  - Due: Sat, Oct. 12 at 11:59pm
- Last chance to switch between 10-418 / 10-618 is October 7th (drop deadline)
- Today's after-clas office hours are un-cancelled (i.e. I am having them)

MBR DECODING

Minimum Bayes Risk Decoding

- Suppose we given a loss function *l(y', y)* and are asked for a single tagging
- How should we choose just one from our probability distribution *p(y|x)*?
- A minimum Bayes risk (MBR) decoder *h(x)* returns the variable assignment with minimum **expected** loss under the model's distribution

$$
\begin{aligned}
h_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \operatorname*{argmin}_{\hat{\boldsymbol{y}}} \; \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot | \boldsymbol{x})} [\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})] \\
&= \operatorname*{argmin}_{\hat{\boldsymbol{y}}} \; \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})
\end{aligned}
$$

Minimum Bayes Risk Decoding

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \; \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot | \boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

Consider some example loss functions:

The **Hamming loss** corresponds to accuracy and returns the number of incorrect variable assignments:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{V} (1 - \mathbb{I}(\hat{y}_i, y_i))$$

The MBR decoder is:

$$\hat{y}_i = h_{\boldsymbol{\theta}}(\boldsymbol{x})_i = \underset{\hat{y}_i}{\text{argmax}} \; p_{\boldsymbol{\theta}}(\hat{y}_i \mid \boldsymbol{x})$$

This decomposes across variables and requires the variable marginals.

Minimum Bayes Risk Decoding

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \ \mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

Consider some example loss functions:

The ***0-1* loss function** returns *1* only if the two assignments are identical and *0* otherwise:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

The MBR decoder is:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underset{\hat{\boldsymbol{y}}}{\text{argmin}} \ \sum_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})(1 - \mathbb{I}(\hat{\boldsymbol{y}}, \boldsymbol{y}))$$

$$= \underset{\hat{\boldsymbol{y}}}{\text{argmax}} \ p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}} \mid \boldsymbol{x})$$

which is exactly the MAP inference problem!

LINEAR PROGRAMMING & INTEGER LINEAR PROGRAMMING

Linear Programming

Whiteboard

- Example of Linear Program in 2D
- LP Standard Form
- Converting an LP to Standard Form
- LP and its Polytope
- Simplex algorithm (tableau method)
- Interior points algorithm(s)

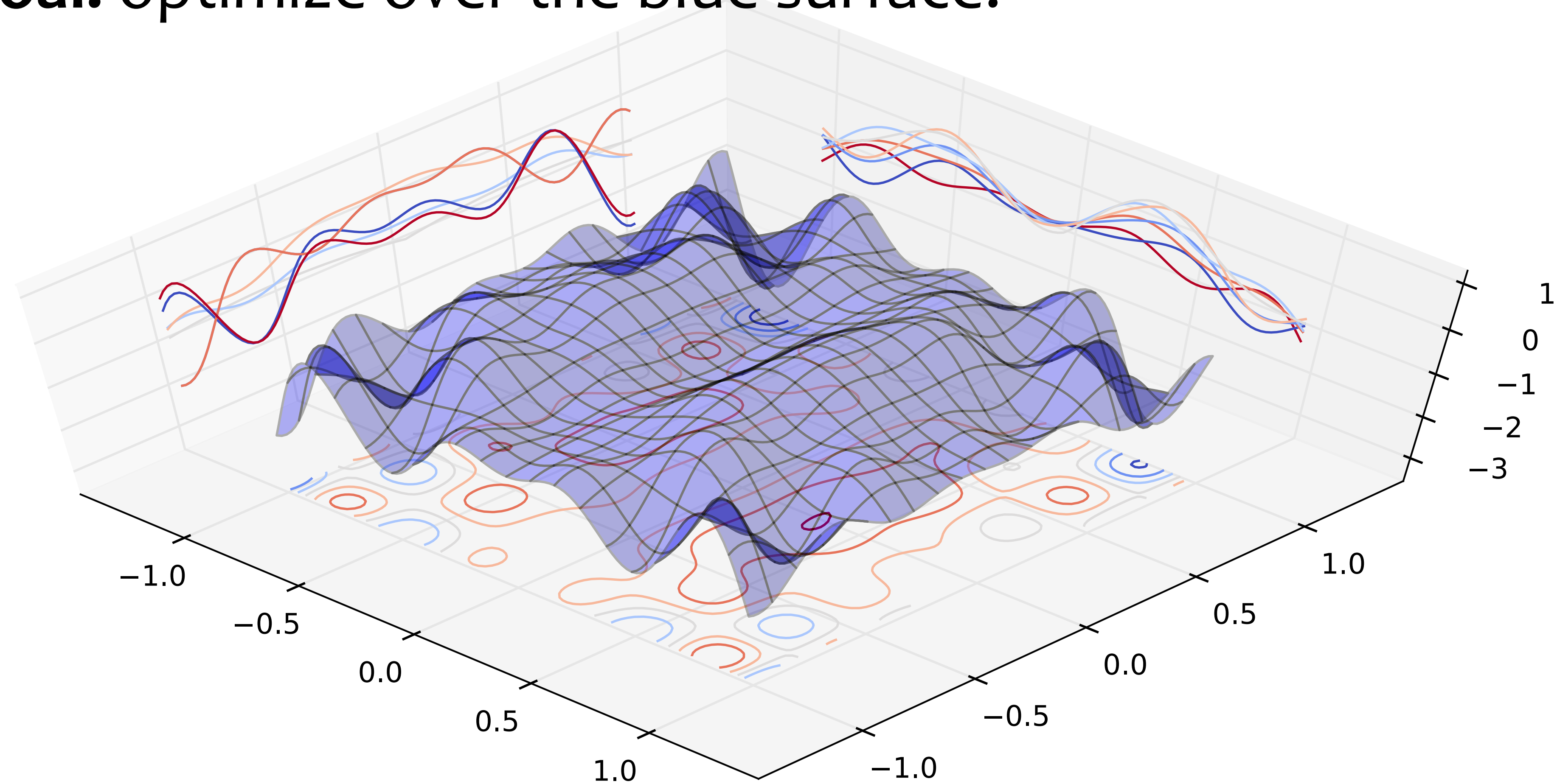
Integer Linear Programming

Whiteboard

- Example of an ILP in 2D
- Example of an MILP in 2D

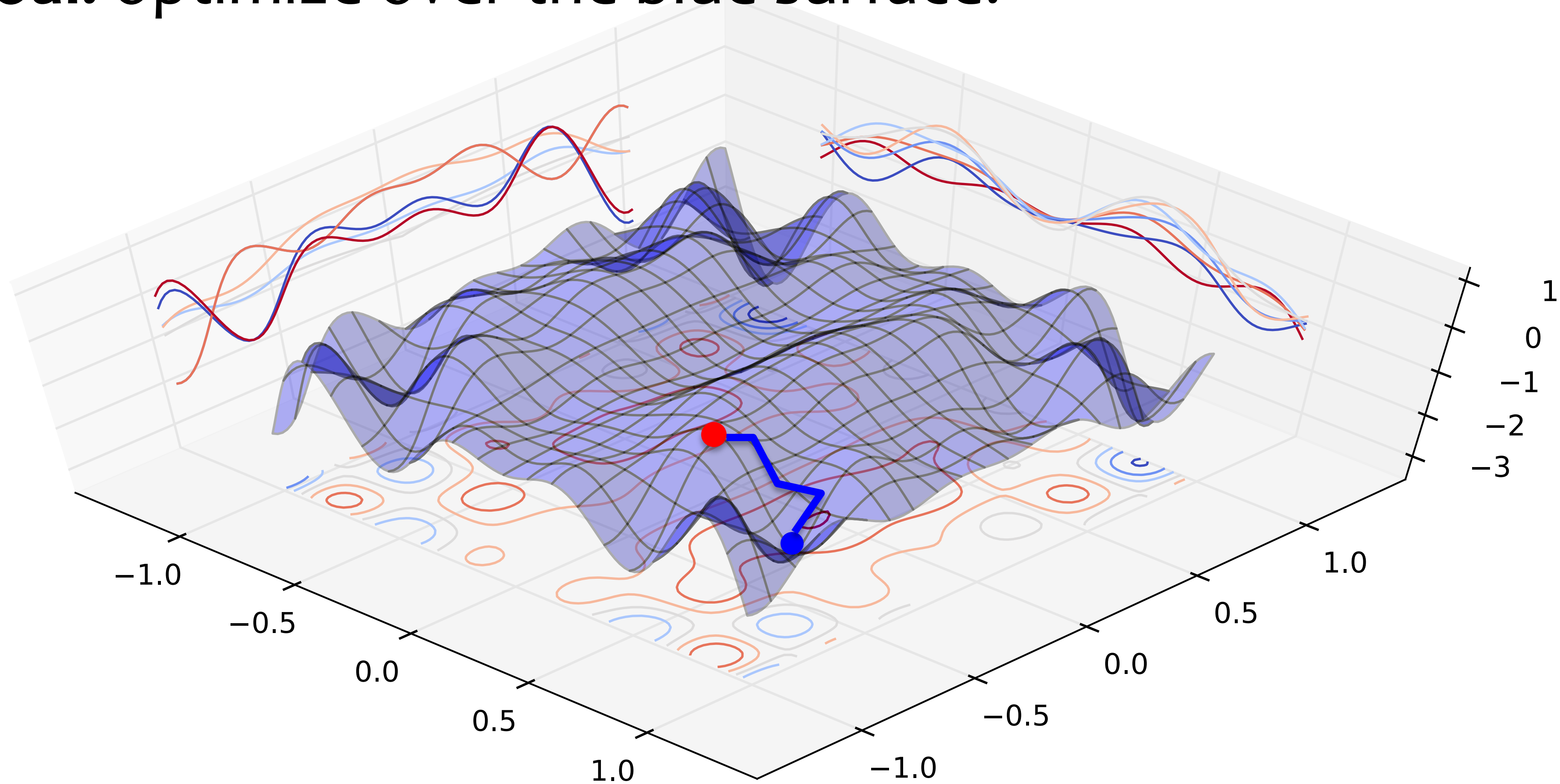
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.



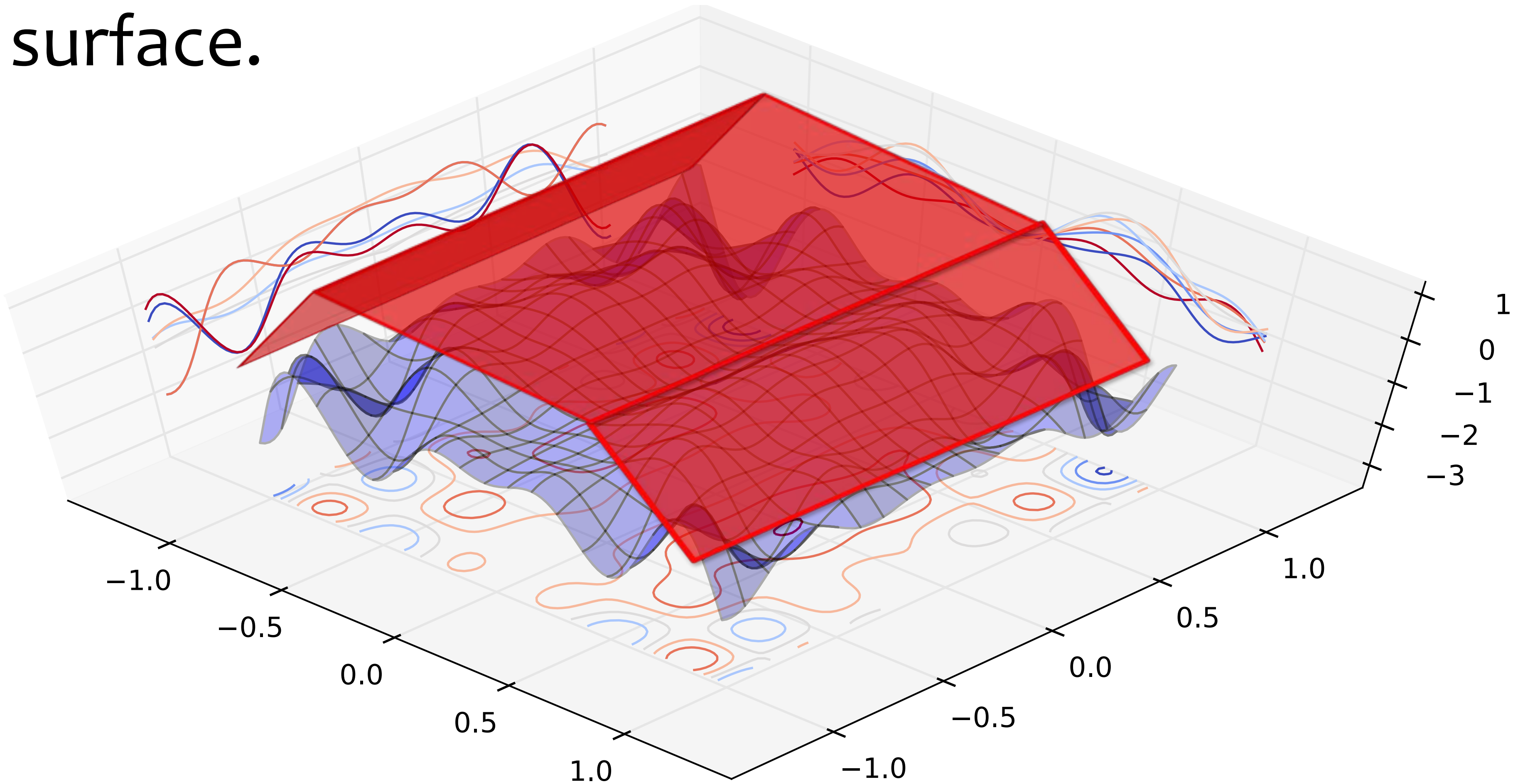
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.



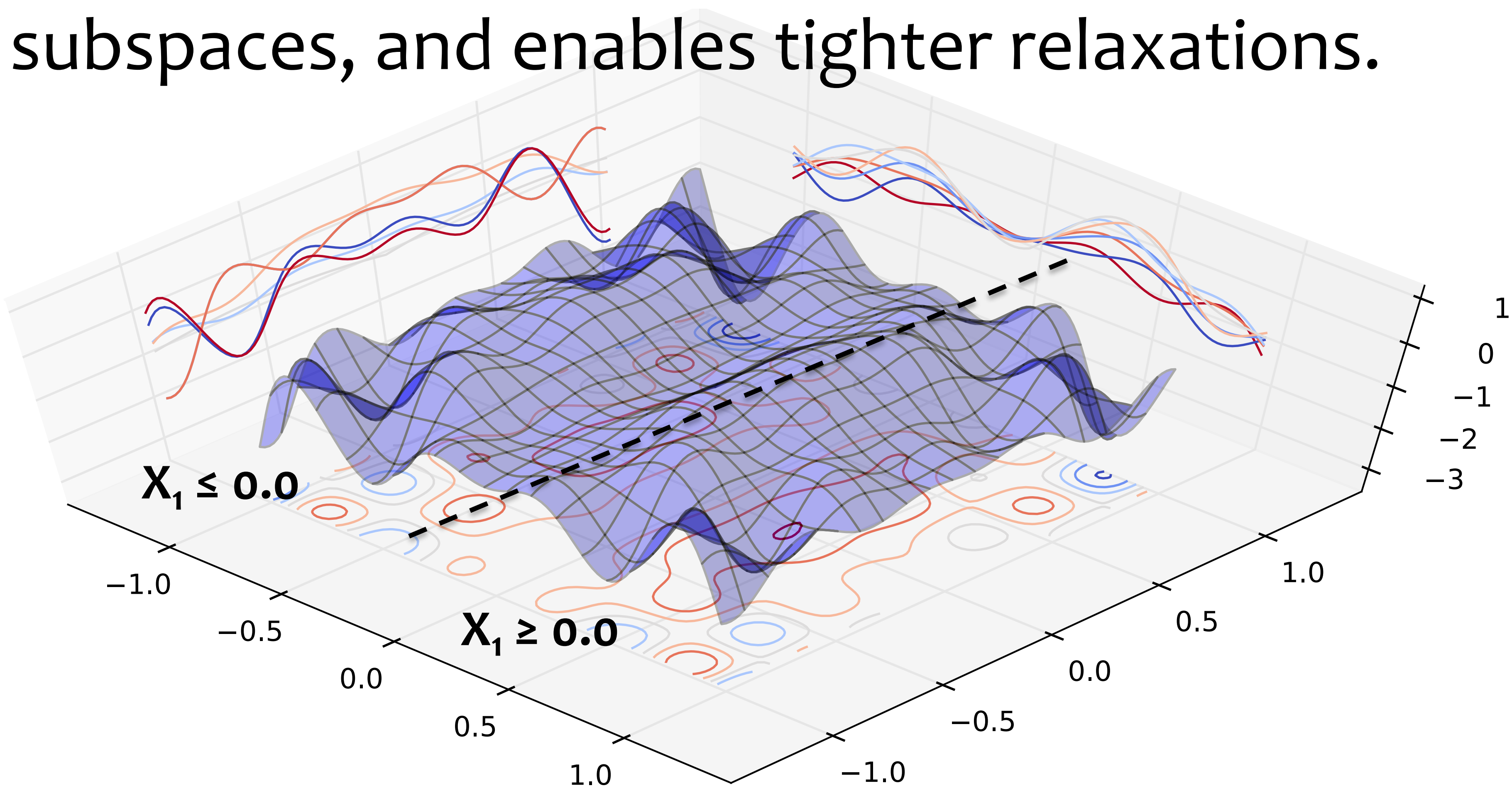
Background: Nonconvex Global Optimization

Relaxation: provides an upper bound on the surface.



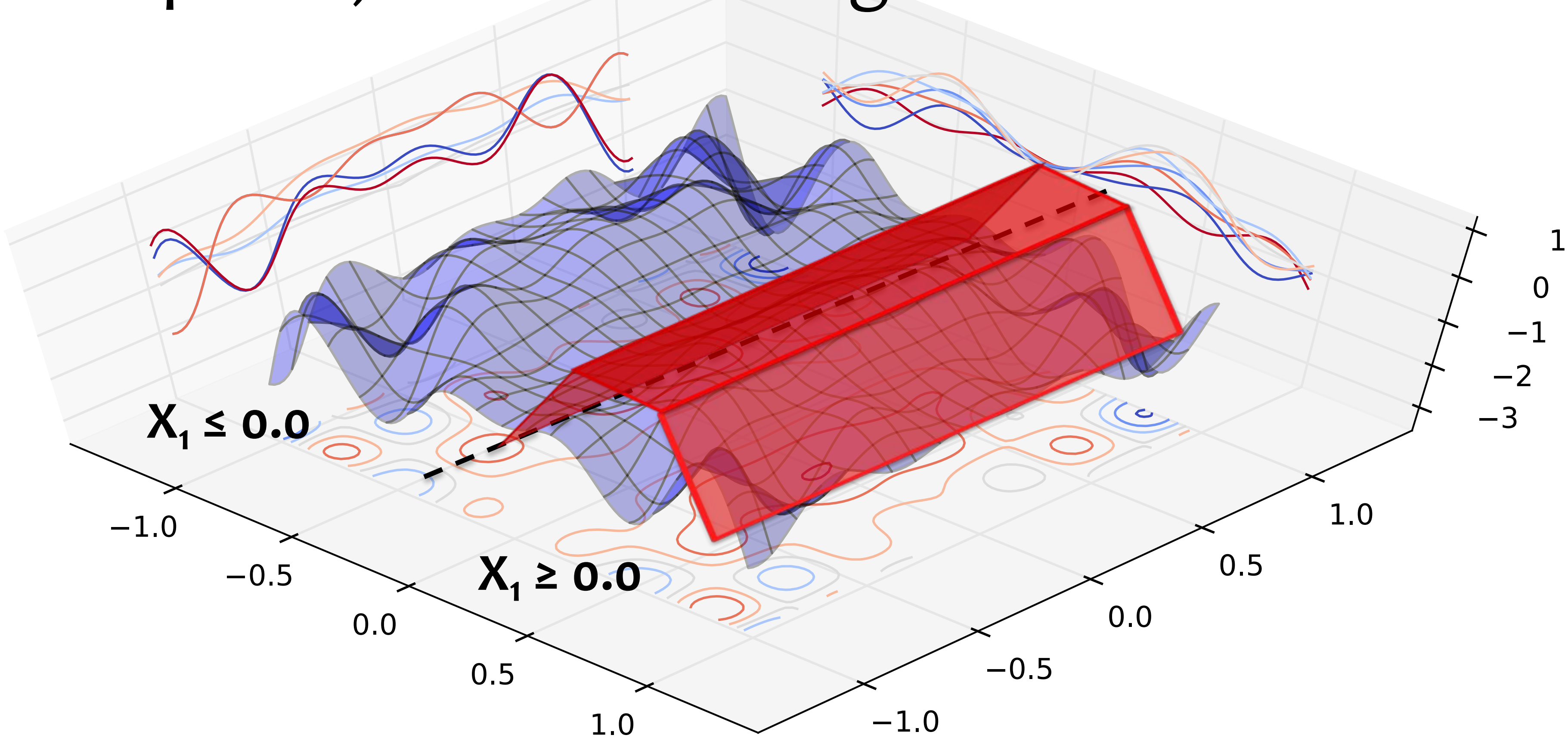
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



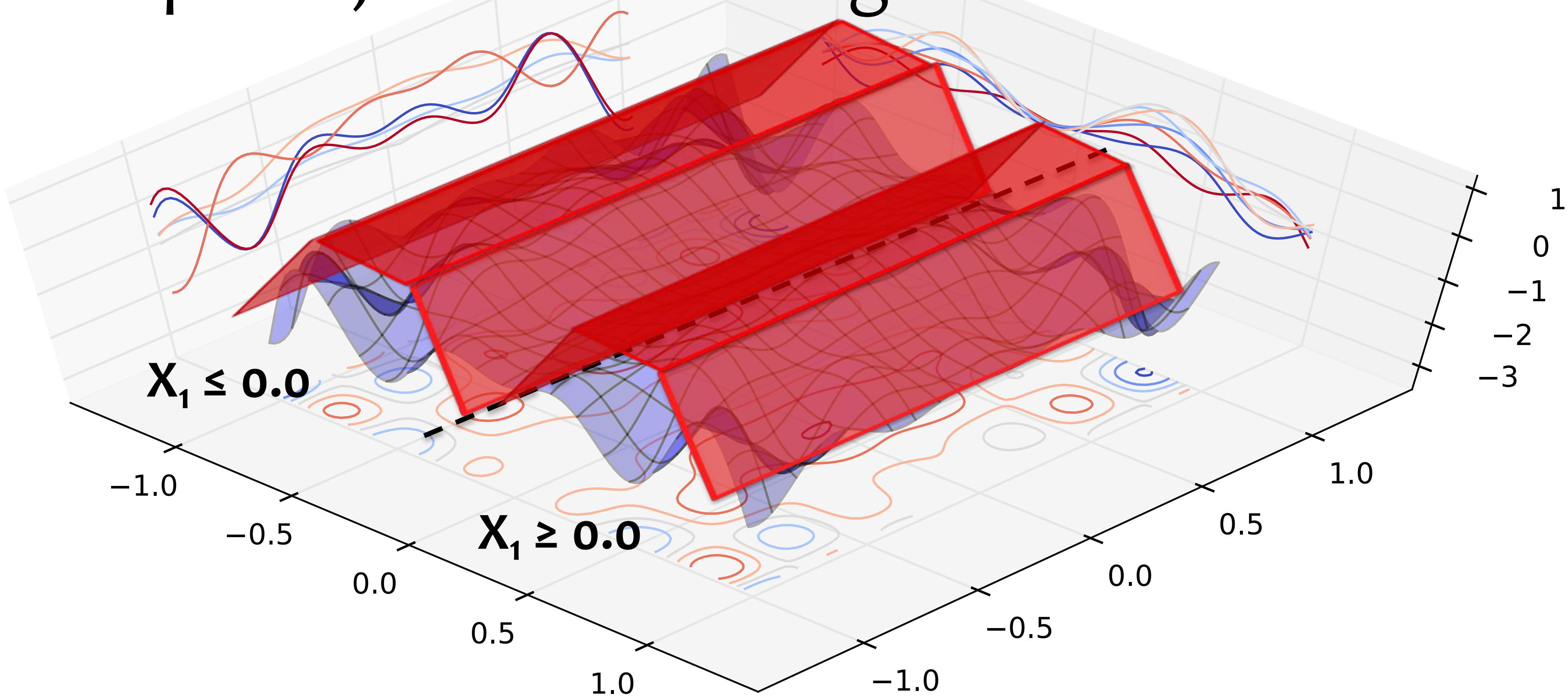
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



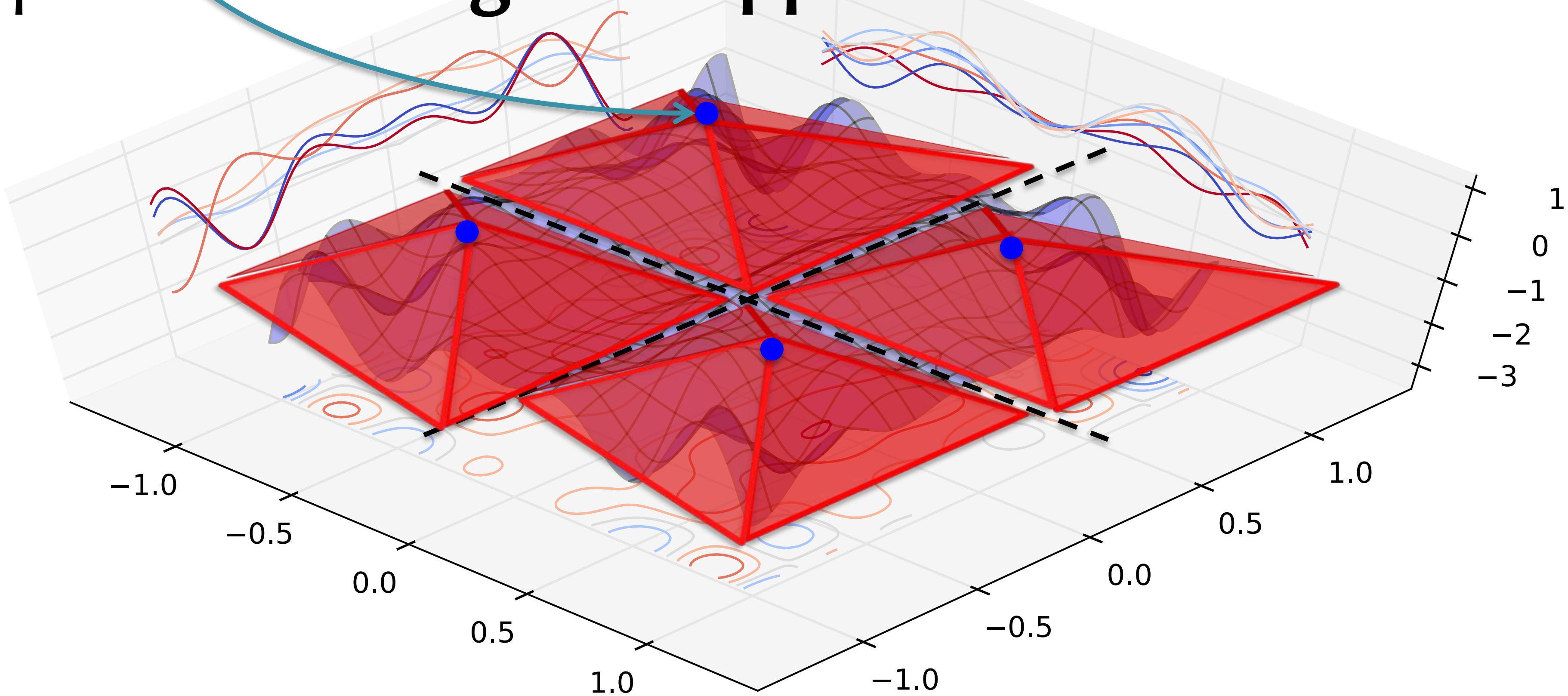
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



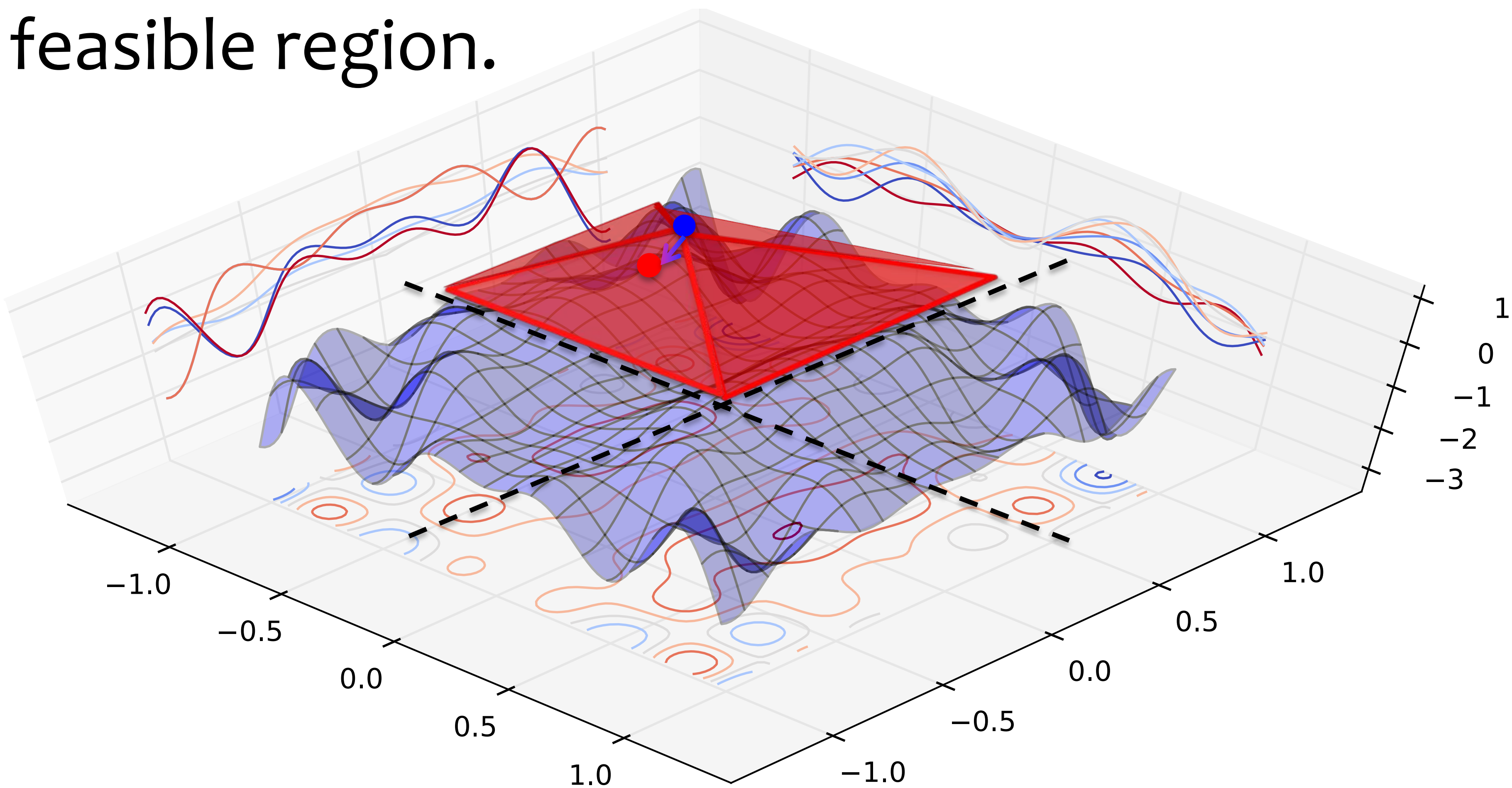
Background: Nonconvex Global Optimization

The **max** of all relaxed solutions for each of the partitions is a **global upper bound.**



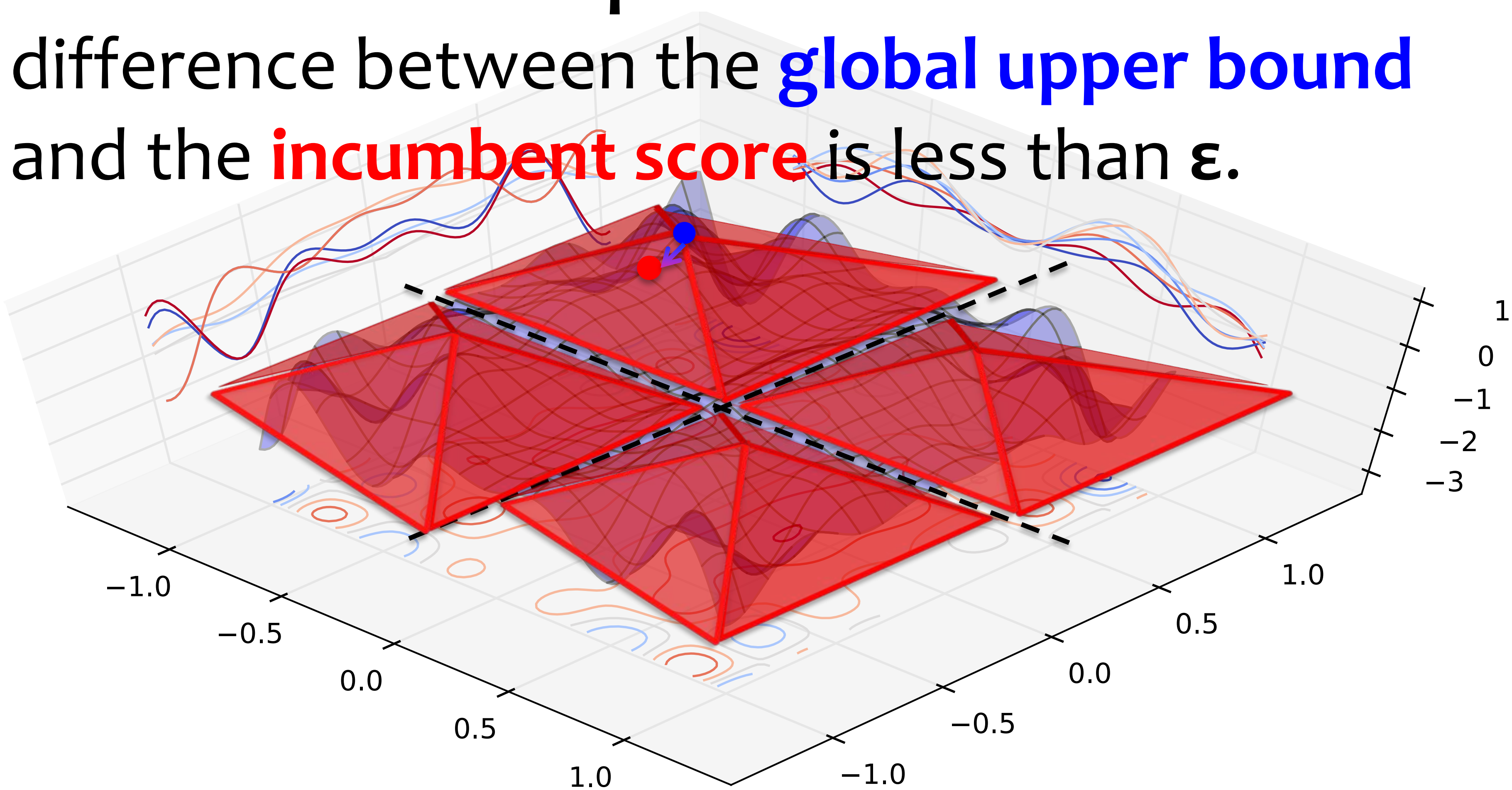
Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.



Background: Nonconvex Global Optimization

The **incumbent** is **ε-optimal** if the relative difference between the **global upper bound** and the **incumbent score** is less than **ε.**



How much should we subdivide?

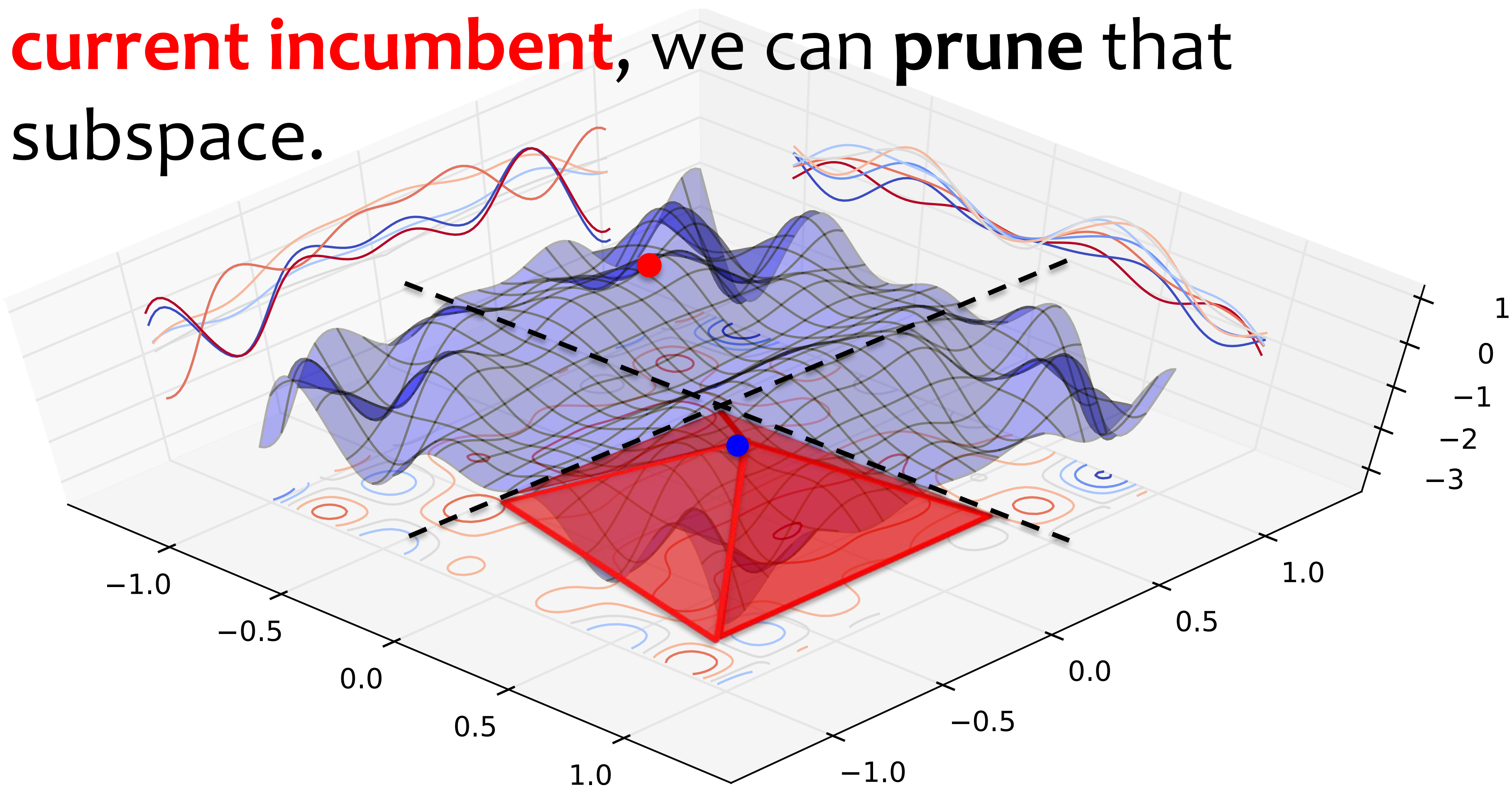
How much should we subdivide?

BRANCH-AND-BOUND

- Method for **recursively subdividing** the search space
- **Subspace order** can be determined heuristically (e.g. best-first search with depth-first plunging)
- **Prunes** subspaces that can't yield better solutions

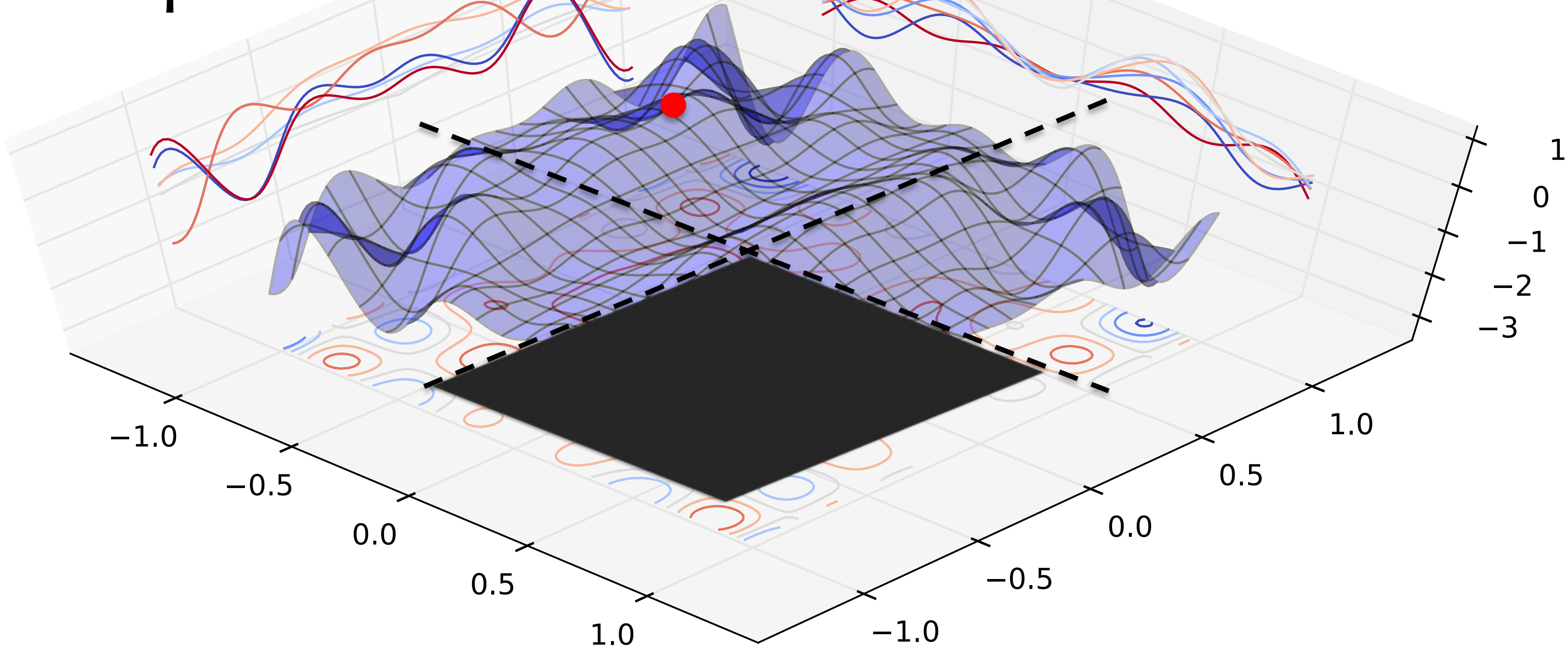
Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.



Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.



Limitations:

Branch-and-Bound for the Viterbi Objective

- The Viterbi Objective
  - Nonconvex
  - NP Hard to solve (Cohen & Smith, 2010)
- Branch-and-bound
  - Kind of tricky to get it right…
  - Curse of dimensionality kicks in quickly
    - Nonconvex quadratic optimization by LP-based branch-and-bound usually fails with more than 80 variables (Burer and Vandenbussche, 2009)
    - Our smallest (toy) problems have hundreds of variables
- Preview of Experiments
  - We solve 5 sentences, but on 200 sentences, we couldn't run to completion
  - Our (hybrid) global search framework incorporates local search
  - This hybrid approach sometimes finds higher likelihood (and higher accuracy) solutions than pure local search

BRANCH-AND-BOUND INGREDIENTS

Mathematical Program

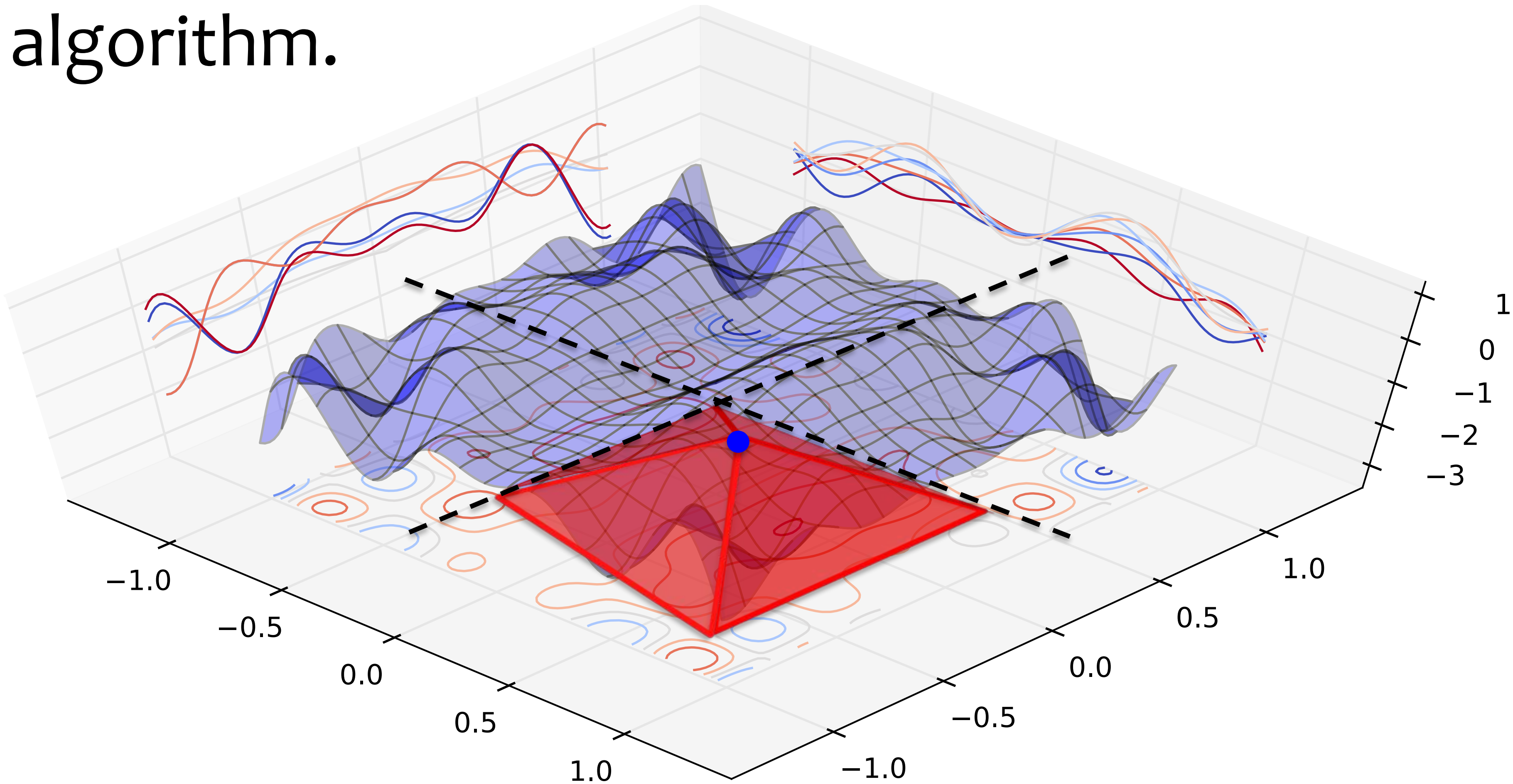
Relaxation

Projection

(Branch-and-Bound Search Heuristics)

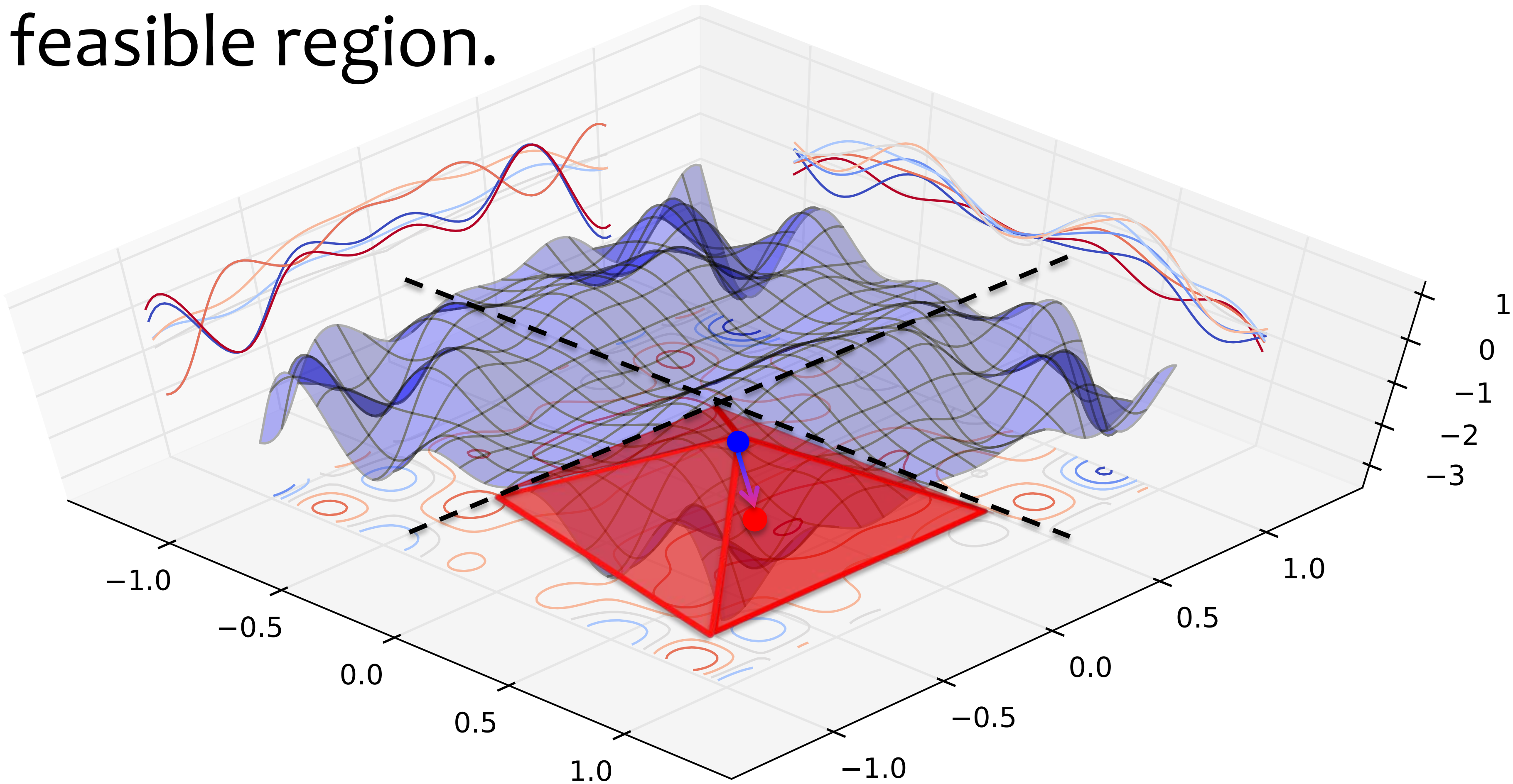
Background: Nonconvex Global Optimization

We solve the **relaxation** using the Simplex algorithm.



Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.



Integer Linear Programming

Whiteboard

– Branch and bound for an ILP in 2D

Branch and Bound

Algorithm 2.1 Branch-and-bound

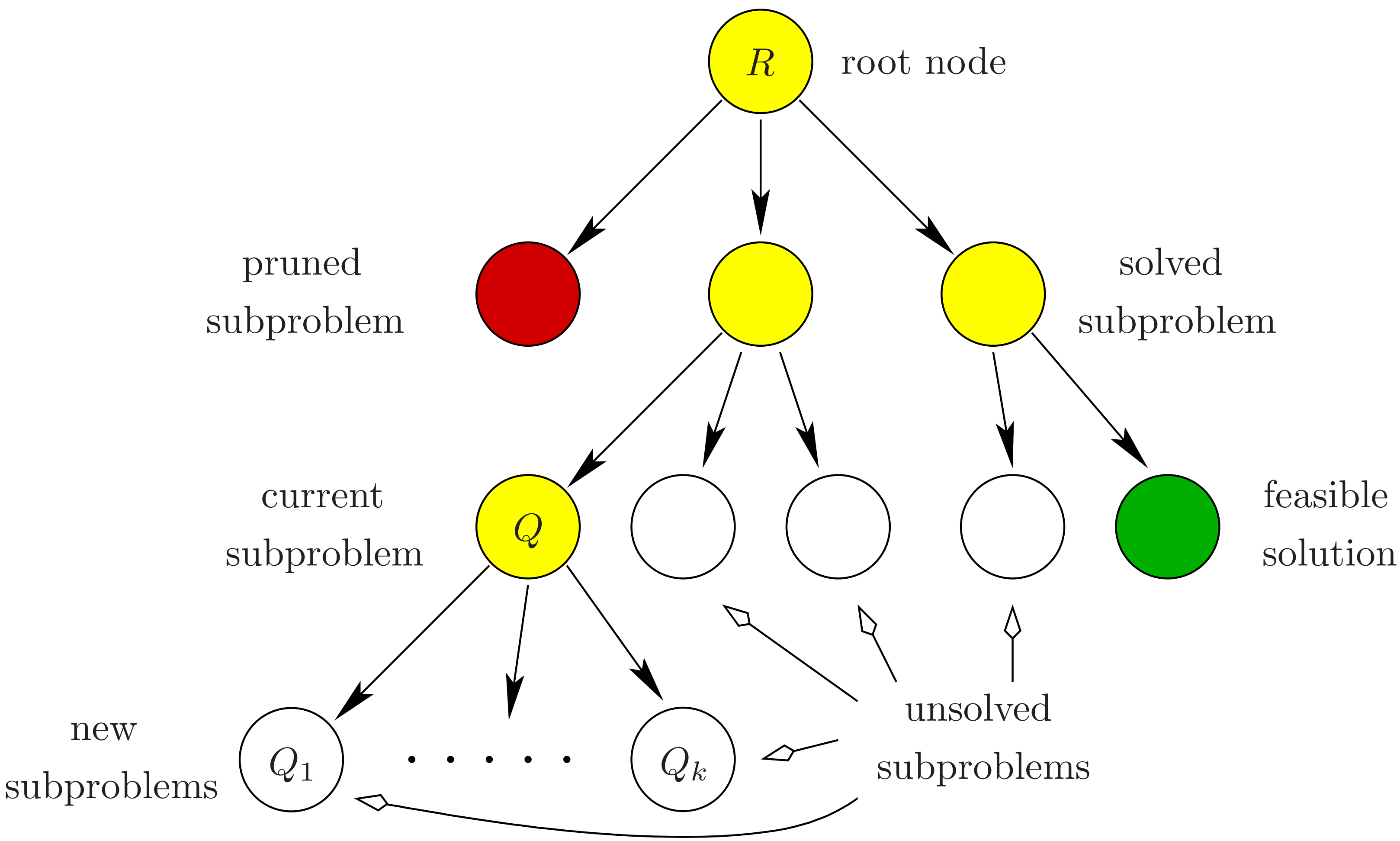
Input: Minimization problem instance R.

Output: Optimal solution $x^\star$ with value $c^\star$, or conclusion that R has no solution, indicated by $c^\star = \infty$.

1. Initialize $\mathcal{L} := \{R\}$, $\hat{c} := \infty$. [*init*]
2. If $\mathcal{L} = \emptyset$, stop and return $x^\star = \hat{x}$ and $c^\star = \hat{c}$. [*abort*]
3. Choose $Q \in \mathcal{L}$, and set $\mathcal{L} := \mathcal{L} \setminus \{Q\}$. [*select*]
4. Solve a relaxation Q_{relax} of Q. If Q_{relax} is empty, set $\check{c} := \infty$. Otherwise, let $\check{x}$ be an optimal solution of Q_{relax} and $\check{c}$ its objective value. [*solve*]
5. If $\check{c} \geq \hat{c}$, goto Step 2. [*bound*]
6. If $\check{x}$ is feasible for R, set $\hat{x} := \check{x}$, $\hat{c} := \check{c}$, and goto Step 2. [*check*]
7. Split Q into subproblems $Q = Q_1 \cup \ldots \cup Q_k$, set $\mathcal{L} := \mathcal{L} \cup \{Q_1, \ldots, Q_k\}$, and goto Step 2. [*branch*]

Slide from Achterberg (thesis, 2007)

Branch and Bound



Slide from Achterberg (thesis, 2007)

Branch and Bound

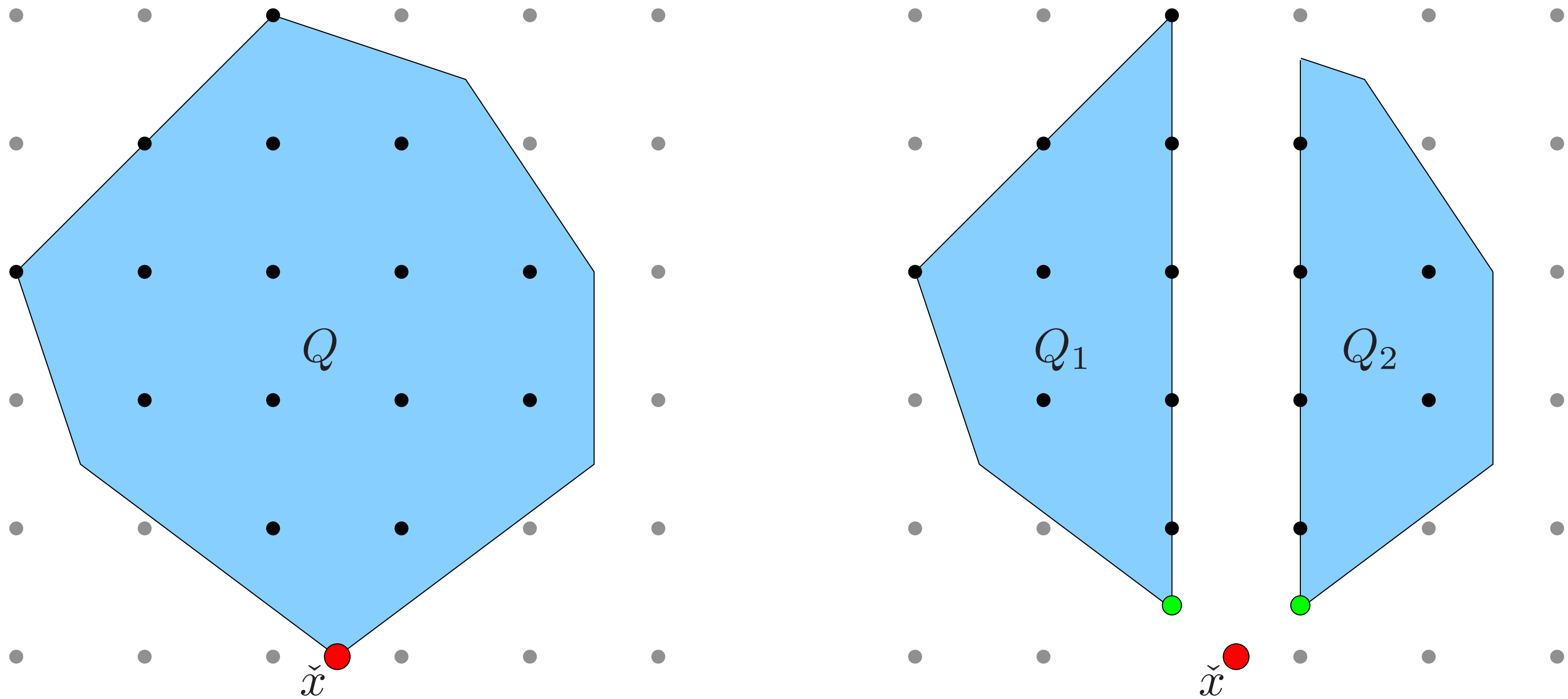


Figure 2.2. LP based branching on a single fractional variable.

Slide from Achterberg (thesis, 2007)